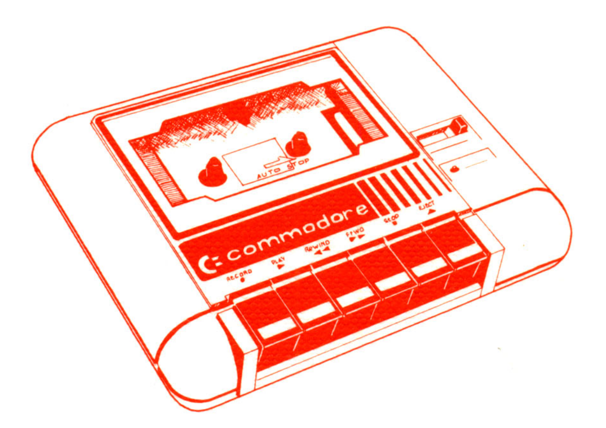
Paulissen

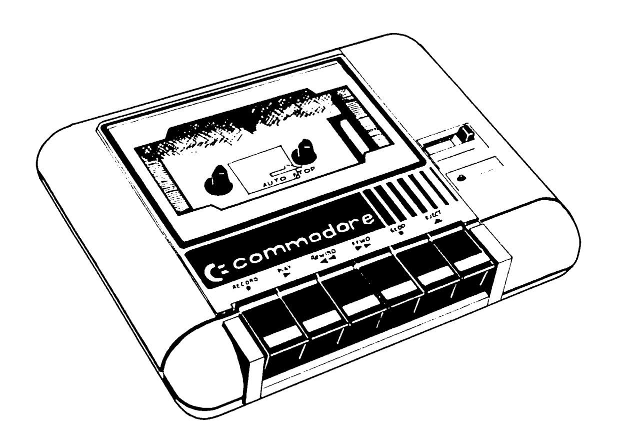
DAS CASSETTEN BUCH ZU COMMODORE 64 UND VC-20



EIN DATA BECKER BUCH

Paulissen

DAS CASSETTEN BUCH ZU COMMODORE 64 UND VC-20



EIN DATA BECKER BUCH

ISBN 3-89011-030-4

Wichtiger Hinweis

Die in diesem Buch wiedergegebenen Schaltungen, Verfahren und Programme werden ohne Rücksicht auf die Patentlage mitgeteilt. Sie sind ausschließlich für Amateur- und Lehrzwecke bestimmt und dürfen nicht gewerblich genutzt werden.

Alle Schaltungen, technische Angaben und Programme in diesem Buch wurden von dem Autor mit größter Sorgfalt erarbeitet bzw. zusammengestellt und unter Einschaltung wirksamer Kontrollmaßnahmen reproduziert. Trotzdem sind Fehler nicht ganz auszuschließen. DATA BECKER sieht sich deshalb gezwungen, darauf hinzuweisen, daß weder eine Garantie noch die juristische Verantwortung oder irgendeine Haftung für Folgen, die auf fehlerhafte Angaben zurückgehen, übernommen werden kann. Für die Mitteilung eventueller Fehler ist der Autor jederzeit dankbar.

INHALTSVERZEICHNIS

1. EINLEITUNG

Wenn man sich einen Rechner gekauft hat, stellt man bald fest, wie unangenehm es ist, daß der Rechner durch das Ausschalten alles vergißt, was man ihm mit viel Mühe beigebracht hat. Man braucht also eine Möglichkeit, was man dem Rechner beigebracht hat, sicher zu lagern, wenn dieser seine wohlverdiente Ruhepause macht.

Da alles, was der Rechner weiß, aus Nullen und Einsen besteht, liegt es nahe, diese Informationen magnetisch aufzuzeichnen, zum Beispiel auf einer Cassette.

Damit der Rechner aber das, was er einmal abgespeichert hat, wieder lesen kann, muß er es mit einem festen Format abspeichern, er muß dem Programm bestimmte Parameter mitgeben. Um das zu gewährleisten, ist im Rechner ein Programm fest eingebaut, das dies alles regelt.

Wenn sie aber Ihre Bedienungsanleitung zur Hand nehmen, stellen Sie fest, daß dort über die Art und Weise, wie der C-64 oder VC 20 mit der Datassette arbeitet und wie Sie sie bedienen müssen, recht wenig steht.

Dieses Buch nun soll Ihnen helfen, besser mit der Datassette arbeiten zu können und zu verstehen, wie der Computer mit ihr zusammenarbeitet.

Anhand von vielen kleinen Beispielprogrammen möchte ich Ihnen weiterhin nahebringen, daß Sie mit der Datassette bedeutend mehr anfangen können, als das Handbuch vermuten läßt. Als krönenden Abschluß finden Sie im zweiten Teil dieses Buches ein komplettes, neues Betriebssystem zur Daten- und Programmspeicherung mit der Datassette. Dieses System arbeitet um einiges schneller als das eingebaute Betriebssystem, ja sogar schneller als eine Diskettenstation. Mit diesem Programmpaket möchte ich

zeigen, daß für den privaten Personalcomputer der bedeutend preiswertere Cassettenrecorder meistens ausreichend ist, wenn man nur die richtige Software benutzt.

Und nun noch ein Hinweis zur Benutzung dieses Buches:

Alle in diesem Buch veröffentlichen Programme sind sowohl auf dem C-64 als auch auf dem VC 20 lauffähig. Da es nicht bei allen Programmen möglich ist ein Programm zu schreiben, das auf den beiden Computern lauffähig ist, sind zwei Versionen abgedruckt.

Bei Maschinensprache-Programmen sind jeweils ein ASSEMBLER-Listing, ein Basic-Lader für den C-64 und ein Basic-Lader für den VC 20 abgedruckt, um eine möglichst große Benutzerfreundlichkeit zu erreichen.

In den Basic-Programmen sind zur besseren Lesbarkeit alle Steuerzeichen übersetzt worden und in eckige Klammern gesetzt worden. Wenn Sie also HOME,RVSON in eckigen klammern lesen, müssen Sie die CLR/HOME-Taste drücken für HOME und dann die CTRL-Taste mit der Taste 9 für RVSON.

TEIL 1

2. DIE BEFEHLE ZUR DATASSETTEN-HANDHABUNG

Das Commodore-Basic stellt Ihnen acht Befehle zum Arbeiten mit der Datassette zur Verfügung:

1. SAVE Programme speichern
2. LOAD Programme laden
3. VERIFY Programme auf Richtigkeit überprüfen
4. OPEN Eine Datei eröffnen
5. CLOSE Eine Datei schließen
6. PRINT# Einen Ausdruck senden
7. INPUT# Einen Ausdruck einlesen
8. GET# Ein einzelnes Byte einlesen

Im folgenden möchte ich die einzelnen Befehle näher beschreiben.

2.1 SAVE

Mit diesem Befehl veranlassen Sie den Computer, das im Speicher befindliche Programm abzuspeichern. Um zu wissen, an welcher Stelle das Programm im Speicher steht, gibt es in Ihrem Rechner einige Speicherstellen, in denen steht, wo das Programm beginnt und wo es endet.

Der Rechner legt sich nämlich sofort nach dem Einschalten ein "Merkheft" an. Dieses Heft hat vier Seiten (0 - 3), sogenannte "Pages" und liegt am Anfang des ganzen Speicherbereiches. Jeweils 256 Speicherstellen werden aufgrund der prozessoreigenen Speicherverwaltung zu einer Page zusammengefaßt, da jede Speicherstelle oder auch jedes Byte genau 256 (0 - 255) verschiedene Werte annehmen kann. In der nullten Seite, also von 0 bis 255, merkt sich der Rechner die Werte, die er sehr oft braucht.

BASIC RAM Aufteilung und Zeiger

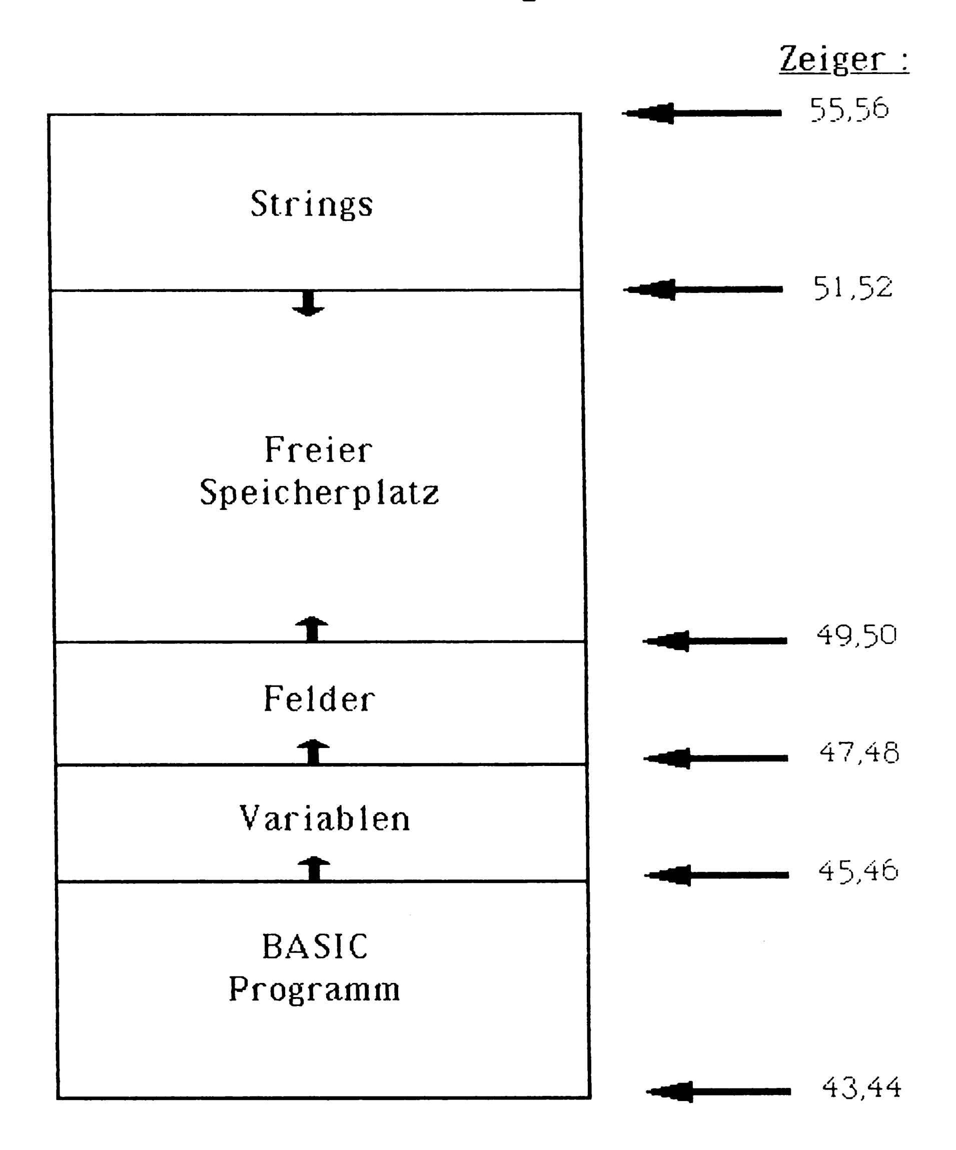


Bild 1 Allgemeiner Aufbau und Zeiger des BASIC RAM

In den Stellen 43, 44 steht immer der aktuelle Start des Basic-Speichers und in 45, 46 das Ende eines Basic-Programms.

Schauen Sie doch einmal nach. Geben Sie PRINT PEEK (43), PEEK (44), PEEK (45), PEEK (46) ein. Auf dem Bildschirm erscheint dann

```
1         8         3         8
```

wenn sie kein Programm geladen haben. Die Bedeutung der einzelnen Zeiger können Sie der Abbildung 1 entnehmen.

Da der Rechner 65535 Speicherstellen hat, werden Adressen immer durch 2 Bytes ausgedrückt. Das erste Byte, das LSB (Last Significant Byte), gibt die Position innerhalb einer Seite an, das zweite, das MSB (Most Significant Byte), die Seitennummer.

Unser Basic-Bereich fängt also auf Seite 8 an der ersten Speicherstelle an, also mit der Speicherstelle 8*256+1 = 2049.

"Wieso stehen aber in den Speicherstellen 45, 46 die Werte 3 und 8, ich habe doch gar kein Programm im Speicher ?", werden Sie jetzt fragen. Das hat folgenden Grund: Damit der Rechner weiß, wo ein Basicprogramm zu Ende ist, hängt er immer 2 Nullen an das Ende, um es zu markieren. Sie haben also jetzt ein Programm im Speicher, welches sozusagen nur aus einer Programmendmarkierung besteht. Sie können dies leicht überprüfen. Wenn Sie

```
PRINT PEEK (PEEK(45)+PEEK(46)*256 -1)
```

eingeben, erhalten Sie 0. Geben Sie jetzt einmal eine Basic-Zeile ein, z.B.

```
10 PRINT "HALLO"
```

und schließen Sie die Eingabe mit RETURN ab. Wenn Sie jetzt

wie oben die Inhalte der Speicherstellen 43, 44, 45, 46 auslesen, erhalten Sie :

1 8 16 8

Der Rechner hat entsprechend Ihres Programmes die Eintragungen in seinem Merkheft geändert.

Nach dieser Exkursion in die Speicherverwaltung kehren wir wieder zum SAVE-Befehl zurück.

Durch diesen Befehl veranlassen wir also den Rechner, alles, was zwischen den eben genannten Zeigern steht, auf Band abzuspeichern. Dem Computer ist es dabei egal, ob auf dem Band schon etwas abgespeichert ist oder nicht. Wie bei einem normalen Cassettenrecorder wird die Cassette einfach überschrieben.

Damit Sie aber das abgespeicherte Programm wieder finden können, haben Sie die Möglichkeit, dem abzuspeichernden Programm einen Namen zu geben. Die Syntax ist dann folgende :

```
SAVE"NAME"
```

Sie haben natürlich auch die Möglichkeit, den String "NAME" vorher in eine Stringvariable einzulesen und ein Programm mit folgender Syntax abzuspeichern :

```
A$="NAME"
SAVEA$
```

Diese Möglichkeit kann wichtig sein, wenn Sie diesen Befehl innerhalb von Programmen nutzen. Aber darüber finden Sie im Kapitel 5 mehr.

Nach dem Namen haben Sie dann noch die Möglichkeit, eine Geräteadresse und eine Sekundäradresse, jeweils durch Komma getrennt, folgen zu lassen. Die genaue Bedeutung der Sekundäradresse finden Sie in Kapitel 3 beschrieben.

Die Geräteadresse sagt dem Computer, welches Peripheriegerät er wählen soll. Sie stellt sozusagen die Hausnummer der verschiedenen Geräte dar. Die Hausnummer der Datassette ist 1, die einer Diskettenstation 8 oder 9.

Damit Sie aber nicht soviel tippen müssen, ist Ihr Computer so programmiert, daß er immer die Datassette auswählt, wenn keine Angabe über die Geräteadresse gemacht wird.

2.2 LOAD

Mit diesem Befehl können Sie das einmal Abgespeicherte wieder laden. Geben Sie dem Rechner keinen Namen an, läd er das nächste Programm, welches er finden kann. Sie können bei diesem Befehl genauso wie bei SAVE einen Namen angeben. Auch können Sie hier wieder eine Geräte- und Sekundäradresse folgen lassen. Hierfür gilt das gleiche wie das beim SAVE-Befehl.

Nach jedem LOAD-Befehl im Direktmodus werden vom Rechner automatisch die Eintragungen in seinem Merkheft entsprechend dem geladenen Programm geändert. Dies ist zu beachten, wenn Sie Maschinenprogramme laden. Wenn Sie z.B. ein Maschinenprogramm laden, das am Ende des Basic-Speichers liegt, werden dementsprechend auch die Basic-Endvektoren auf das Basic-RAM-Ende gesetzt. Eine weitere Eingabe wird danach mit "?OUT OF MEMORY ERROR" vom Rechner kommentiert. Um nun weitere Eingaben machen zu können, müssen Sie entweder in die Zeiger 45,46 den alten Wert POKEn oder Sie geben einen NEW-Befehl, damit das im Basic-Speicher befindliche Programm nicht aber das Maschinenprogramm gelöscht wird.

Sie können den NEW-Befehl umgehen, wenn Sie das Maschinenprogramm im Programmodus laden. Näheres lesen Sie dazu im Kapitel 5.

2.3 VERIFY

Dieser Befehl arbeitet ähnlich wie der LOAD-Befehl. Der einzige Unterschied ist, daß durch diesen Befehl das Programm nicht in den Speicher geschrieben wird, sondern Byte für Byte mit dem im Speicher stehenden verglichen wird. Dadurch können Sie testen, ob eine Programmspeicherung erfolgreich war.

War die Speicherung erfolgreich und hat auch die Cassette keinen Fehler, erscheint OK auf dem Bildschirm. Wird bei dem Vergleich ein Fehler festgestellt, erhalten Sie die Meldung "?VERIFY ERROR".

2.4 OPEN

Bis jetzt habe ich nur über das Laden und Speichern von Programmen geschrieben. Sie haben aber auch die Möglichkeit, Daten zu speichern und zu laden. Um dem Rechner mitzuteilen, daß Sie Daten laden oder speichern wollen, müssen Sie mit dem OPEN-Befehl eine Datei eröffnen.

Mit diesem Befehl teilen Sie dem Computer alle Parameter mit, die er benötigt, nämlich die Gerätenummer, den Filenamen und die Sekundäradresse, die aussagt, ob geladen oder gespeichert werden soll. Um eine Datei zu eröffnen, müssen Sie folgenden Befehl eingeben :

```
OPEN lf,GA,SA,"NAME"
```

Mit "NAME" können Sie der Datei wie bei SAVE und LOAD einen Namen geben.

SA steht für Sekundäradresse:

> Über diese Zahl teilen Sie dem Computer mit, ob Sie Daten senden (= 1) oder empfangen (= 0) wollen.

Geben Sie eine 2 ein, wird nach dem Abspeichern noch ein EOT (End Of Tape) Block geschrieben. Siehe dazu auch Kapitel 3.

GA steht für Geräteadresse:

Diese Zahl sagt dem Rechner, mit welchem Gerät er arbeiten soll. Dabei bedeutet

0	= Tastatur
1	= Cassette
3	= Bildschirm
4,5	= Drucker
8,9	= Diskette

lf steht für logische Filenummer:

Diese Zahl, sie kann zwischen 0 und 255 liegen, stellt den Index dieser OPEN - Anweisung dar. Damit nicht bei jedem Ein- und Ausgabebefehl alle Parameter mit übergeben werden müssen, legt sich der Computer eine Tabelle an, in der die Parameter der OPEN - Anweisung unter dieser Indexnummer abgelegt sind.

Nachdem wir nun eine Datei eröffnet haben, kommen wir zu den Befehlen der Daten Ein- und Ausgabe.

2.5 PRINT# UND CLOSE

Mit dem PRINT#-Befehl ist es möglich, einzelne Daten auf Band zu schreiben.

Zur besseren Erklärung geben Sie einmal folgendes Programm ein:

```
10 OPEN 1,1,1,"DATENFILE"
20 PRINT "DATEI WURDE EROEFFNET"
30 PRINT:PRINT"GEBEN SIE DATEN EIN"
40 INPUT"DATEN";A$
50 PRINT#1,A$
60 A=A+1
70 PRINT A"DATEN WURDEN ZWISCHENGESPEICHERT"
80 PRINT:PRINT"WEITERE DATEN"
90 GET F$: IF F$ = "" THEN 90
100 IF F$ = "J" THEN 40
110 CLOSE1
120 PRINT:PRINT"DATEI WURDE GESCHLOSSEN"
130 END
```

Legen Sie nun eine leere Cassette in Ihren Recorder und starten das Programm mit RUN. Es erscheint sofort die Meldung "PRESS RECORD & PLAY ON TAPE". Wenn Sie dieser Meldung Folge leisten, wird entsprechend der OPEN-Anweisung in Zeile 10 ein Dateikopf auf Band geschrieben, der den Namen der Datei enthält. Danach hält der Recorder an und Sie werden durch Zeile 40 gebeten, Daten einzugeben.

In der Zeile 50 wird dann durch den PRINT#-Befehl der eingegebene String zwischengespeichert. Die Nummer nach der PRINT#-Anweisung, es ist die logische Filenummer, stellt den Bezug zur OPEN-Anweisung her. Daß der String nicht sofort auf Band geschrieben wird, können Sie daran sehen, daß sich das Band noch nicht bewegt.

Erst wenn Sie eine ganze Reihe von Daten eingegeben haben, setzt sich das Band in Bewegung, und alle bis dahin

eingegebenen Daten werden auf das Band übertragen. Auf den Zwischenspeicher, es handelt sich um den Cassettenpuffer, werde ich noch im Kapitel 7 genauer eingehen.

Wenn Sie nun auf die Abfrage in Zeile 100 mit "J" antworten, können Sie weitere Daten eingeben. Ist der Puffer voll, können Sie feststellen, daß eine Pause entsteht und sich das Band kurzzeitig bewegt. Jetzt wird der Puffer auf Band geschrieben und anschließend mit "Space" (CHR$(32)) beschrieben, um neue Daten aufzunehmen.

Wenn Sie die Eingabe beenden, verzweigt der Rechner zum CLOSE-Befehl in Zeile 110. Die Zahl hinter diesem Befehl ist wieder eine logische Filenummer und sagt dem Computer, daß er die mit der logischen Filenummer 1 geöffnete Datei schließen soll. Das macht er, indem er alle Eintragungen, die er in seinem Merkheft unter dem Index 1 notiert hat, löscht. Der Rechner schreibt nun hinter die letzte Eintragung ein Nullbyte, um das Datenende zu kennzeichnen, und speichert den Puffer auf Band. Es ist also sehr wichtig, eine Datei wieder zu schließen, da sonst die letzten Daten verlorengehen.

Bis jetzt haben wir nur Strings, bzw. den Inhalt von Stringvariablen abgespeichert. Die Frage ist nun:"Wie verarbeitet der Rechner numerische Ausdrücke und Variablen?" Solche Ausdrücke werden abgespeichert, als wären es Strings. D.h. der Rechner führt bei jedem PRINT#-Befehl automatisch einen STR$-Befehl durch. Es ist also das gleiche, ob Sie

```
PRINT#1,55
```

oder

```
A$=STR$(55):PRINT#1,A$
```

schreiben. Aus diesem Grund können Sie später jeden numerisch abgespeicherten Wert auch in eine Stringvariable einlesen.

Eine weitere wichtige Sache ist das Format einer PRINT#-Anweisung, wenn Sie mit einer Anweisung mehrere Ausdrücke abspeichern wollen. Auf dieses Problem möchte ich aber zum besserem Verständnis erst nach dem INPUT#-Befehl kommen.

2.6 INPUT#

Dieser Befehl verhält sich vollkommen analog zum ´normalen´ INPUT-Befehl, nur daß hier nicht die Tastatur als festes Empfangsgerät voreingestellt ist. Wenn Sie z.B. mit

```
OPEN1,0
```

(0=Tastatur) die Tastatur als Sender definieren, verhält sich INPUT#1,A$ genauso wie INPUTA$, mit dem kleinen Unterschied, daß kein Fragezeichen ausgedruckt wird.

Wie sich nun INPUT# in Hinsicht auf den Cassettenrecorder verhält, sehen wir uns am besten wieder mit einem kleinen Beispielprogramm an. Spulen Sie Ihre Cassette zu dem Punkt zurück, wo Sie die Datei "DATENFILE" abgespeichert haben, und geben Sie folgendes Programm ein :

```
10 OPEN 1,1,0,"DATENFILE"
20 PRINT "DATEI WURDE EROEFFNET"
30 PRINT:PRINT"DATEN WERDEN EINGELESEN"
40 INPUT#1,A$
50 PRINTA$
60 A=A+1
70 PRINT A"DATEN WURDEN EINGELESEN"
80 PRINT:PRINT"WEITERE DATEN"
90 GET F$: IF F$ = "" THEN 90
100 IF F$ = "J" THEN 40
110 CLOSE1
120 PRINT:PRINT"DATEI WURDE GESCHLOSSEN"
130 END
```

Wenn Sie nun das Programm starten und laut Anweisung die PLAY-Taste drücken, sehen Sie, daß das Band einige Zeit läuft und dann anhält. Ihr Rechner hat nun aufgrund des OPEN-Befehls die Datei "DATENFILE" gesucht und dann aufgrund des INPUT#-Befehls in Zeile 40 den ersten Datenblock eingelesen und den ersten String der Variablen A$ übergeben, die über Zeile 50 auf dem Bildschirm angezeigt wird.

Die einzelnen Daten stehen jetzt genauso im Puffer, wie Sie sie beim Abspeichern hereingeschrieben haben. Hinter jedem String steht aufgrund der PRINT#-Anweisung ein 'Carriage Return' (CHR$(13)), also ein Zeilenvorschub. Der PRINT#-Befehl schreibt die Daten genauso in den Puffer wie ein PRINT-Befehl sie auf den Bildschirm schreiben würde. Der INPUT#-Befehl erkennt genauso wie der INPUT-Befehl durch ein Komma oder RETURN, daß hier der Ausdruck zu Ende ist.

Byte	0	1	2	3	4	5	6	7	8	9	10	11	...	27	28	29	30	31
Inhalt	S	T	R	1	CR	S	T	R	2	CR	S	T	...	R	5	CR	00	

Beantworten Sie jetzt die Frage, ob weitere Daten eingelesen werden sollen. Bei unserem Beispielprogramm gebben Sie 'J' ein. Jetzt können Sie String für String wieder einlesen und auf dem Bildschirm ausdrucken lassen, bis Sie die Meldung erhalten "?STRING TOO LONG ERROR".

Diese Meldung erhalten Sie, weil Sie inzwischen den letzten, abgespeicherten String eingelesen haben und nun versuchen, einen weiteren einzulesen. Der Computer sucht das Ende des Strings, also ein Komma oder CARRIAGE RETURN (kurz CR), findet aber in den nächsten folgenden 80 Speicherstellen keins. (Sie können maximal 80 Zeichen mit einer INPUT-bzw. INPUT#-Anweisung einlesen.) Daraus schließt der Computer, daß hier ein String länger als 80 Zeichen vorliegt und gibt die Fehlermeldung aus.

Damit Sie aber nicht bei einem 'normalen' Dateiprogramm immer diese Fehlermeldung bekommen, gibt es eine

Möglichkeit, das Dateiende mit Hilfe der Statusvariablen zu erkennen. Wie Sie das programmieren müssen, finden Sie im Kapitel 4.

Diese Fehlermeldung erhalten Sie aber nicht nur, wenn Sie über das Dateiende hinaus zu lesen versuchen, sondern auch dann, wenn die Daten in einem falschen Format abgespeichert wurden.

Bei der Besprechung des PRINT#-Befehls haben wir immer nur einen Ausdruck pro PRINT#-Anweisung abgespeichert, was zur Folge hatte, daß automatisch immer ein CR angehängt wurde. Es ist aber auch möglich, mehrere Ausdrücke mit einer PRINT# - Anweisung abzuspeichern. Dabei muß aber darauf geachtet werden, daß die einzelnen Ausdrücke voneinander separiert werden. Schreiben Sie z. B.

```
PRINT#1,A$,B$
```

mit A$="STRING1" und B$="STRING2", werden die beiden Ausdrücke genauso in den Puffer geschrieben, wie sie mit einer PRINT-Anweisung auf den Bildschirm geschrieben würden, nämlich

Byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Inhalt	S	T	R	I	N	G	1				S	T	R	I	N	G	2	CR

Wenn Sie aber folgende Anweisungen geben:

```
T$=","
PRINT#1,A$;T$;B$
```

erhalten Sie

Byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Inhalt	S	T	R	I	N	G	1	,	S	T	R	I	N	G	2	CR		

Jetzt sind die Strings eindeutig voneinander getrennt und werden von einer INPUT#1,A$,B$-Anweisung richtig eingelesen.

Auf eine beachtenswerte Besonderheit muß ich aber noch hinweisen. Das Komma reicht zwar aus, um die Daten für eine eindeutige Variablenzuweisung mit INPUT# zu gewährleisten, aber es reicht nicht aus, dem Computer zu sagen, daß hier eine Eintragung zu Ende ist und daß ab dem nächsten Byte eine neue beginnt. Das klingt zwar sehr ähnlich, ist es aber nicht. Deshalb möchte ich im folgenden etwas näher darauf eingehen.

Man kann den Computer mit einer großen Firma mit vielen Abteilungen vergleichen. Das Verarbeiten von Daten beim Lesen und Schreiben überläßt er zwei Abteilungen.

1. Abteilung - Variablenverarbeitung
   Diese Abteilung sorgt dafür, daß ein Ausdruck, der irgendwo im Speicher steht (Cassettenpuffer, Bildschirmspeicher, Programm etc.), der richtigen Variablen im richtigen Format zugeordnet wird.

2. Abteilung - Speicherplatzverwaltung
   Diese Abteilung sagt der ersten Abteilung immer, ab welchem Speicherplatz der für sie interessante Ausdruck steht.

Jede der beiden Abteilung hat nun ihre eigene Art, das Ende eines Ausdrucks und somit den Beginn eines neuen zu suchen. Der ersten Abteilung ist es egal, ob sie ein Komma oder einen CR findet. Bei beiden weiß sie, daß hier ein Ausdruck zu Ende ist und danach ein neuer beginnt. Die zweite Abteilung kümmert sich um das Komma überhaupt nicht. Sie erkennt nur den CR als Trennungszeichen an. D.h. sie gibt als Startpunkt für einen neuen Ausdruck immer die Stelle nach einem CR an.

Am besten veranschaulicht man sich das mit einem kleinen

Beispielprogramm. Geben Sie einmal die folgenden Zeilen ein, legen eine Leercassette in Ihren Recorder und starten das Programm mit RUN.

```
10 OPEN1,1,1,"TEST"
20 T$=","
30 A1$="STRING1" : A2$="STRING2"
40 A3$="STRING3" : A4$="STRING4"
50 PRINT#1,A1$;T$;A2$;T$;A3$
60 PRINT#1,A4$
70 CLOSE1
80 END
100 OPEN1,1,0,"TEST"
110 INPUT#1,A$
120 PRINT A$
130 INPUT#1,A$
140 PRINT A$
150 CLOSE 1
160 END
```

Nachdem die Datei TEST abgespeichert wurde, spulen Sie Ihre Cassette an den Start der Datei zurück und starten das Programm mit RUN100. Wenn der Rechner die Datei eingelesen hat, stehen die Strings folgendermaßen im Speicher:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
S	T	R	I	N	G	1	,	S	T	R	I	N	G	2	,	S	T	R	I

20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
N	G	3	CR	S	T	R	I	N	G	4	CR	00			

Auf dem Bildschirm erhalten Sie:

```
STRING1
STRING4
```

Abteilung zwei hat beim ersten INPUT# seinen Zeiger auf den

ersten String gesetzt, und Abteilung eins hat ihn dann A$ zugeordnet. Da keine weiteren Ausdrücke bei dieser INPUT#-Anweisung eingelesen werden sollen, wird die Ausführung wieder an Abteilung zwei übergeben. Bei der nächsten INPUT#-Anweisung sucht Abteilung zwei das nächste CR und findet es vor STRING4. Diesen Startpunkt übergibt sie der Abteilung eins, welche den Ausdruck "STRING4" wiederum A$ zuweist.

Ändern Sie jetzt einmal Zeile 110 und 120, indem Sie ´,B$,C$´ anhängen. Wenn Sie nun die Datei erneut einlesen, erhalten Sie auf dem Bildschirm :

```
STRING1   STRING2   STRING3
STRING4
```

Jetzt weist Abteilung eins in Zeile 110 erst die beiden folgenden Ausdrücke den Variablen B$ und C$ zu, ehe sie das Kommando an Abteilung zwei zurückgibt.

Kurz :

> Der Zeiger der Abteilung eins wird durch jede Variablenzuweisung auf das nächste Trennungszeichen gesetzt (Komma oder CR).
>
> Der Zeiger der Abteilung zwei wird durch jede INPUT#-Anweisung auf das nächste Trennungszeichen gesetzt (CR).

Zusammenfassend kann man also sagen, daß ein Komma solange als Trennung ausreicht, wie Sie die Daten im gleichen Format einlesen, wie Sie sie abgespeichert haben. Wollen Sie aber beim Einlesen frei sein vom Eingabeformat, dann definieren Sie den Trennungsstring mit CR.

```
20 T$=CHR$(13)
```

Bei so abgespeicherten Daten erhalten Sie immer das gleiche Ergebnis, egal ob Sie mit einer INPUT#-Anweisung eine oder

mehrere Variablen einlesen.

Bis jetzt habe ich nur über die Trennung von einzelnen Ausdrücken geschrieben. Bei manchen Anwendungen kann es aber sinnvoll sein, einzelne Ausdrücke nicht zu trennen. Dies erreicht man wie bei dem PRINT-Befehl auf dem Bildschirm dadurch, daß man eine PRINT#-Anweisung mit einem Semikolon abschließt. Probieren Sie es einmal mit dem kleinen Testprogramm von oben. Ändern Sie die Zeilen folgendermaßen ab:

```
10 OPEN1,1,1,"TEST2"
50 PRINT#1,A1$;A2$;A3$;
100 OPEN1,1,0,"TEST2"
```

Löschen Sie Zeile 130 und 140. Wenn Sie nun mit RUN die Datei TEST2 abspeichern und sie mit RUN100 einladen, erhalten Sie:

```
STRING1STRING2STRING3STRING4
```

Ergeben sich aber durch die Aneinanderreihung Strings, die länger als 80 Zeichen sind, können Sie sie nicht mehr mit der INPUT#-Anweisung einlesen. Solche Strings können Sie dann nur noch Byte für Byte mit der GET#-Anweisung einlesen.

2.7 GET#

Auch dieser Befehl arbeitet vollkommen analog dem GET - Befehl. Mit ihm können Sie byteweise Daten einlesen. Geben Sie einmal folgendes Programm ein:

```
10 OPEN1,1,0,"TEST"
20 GET#1,A$
30 PRINTA$;
40 GETA$: IF A$="" THEN 40
50 GOTO20
```

Spulen Sie nun zum Beginn der Datei TEST zurück und starten Sie das Programm. Nachdem der Computer die Datei gefunden hat, erscheint buchstabenweise folgendes Bild :

```
STRING1   STRING2   STRING3
STRING4
```

Wenn nun noch weitere Zeichen eingelesen werden, können Sie keine Veränderung auf dem Bildschirm erkennen, da der Puffer mit "SPACE" aufgefüllt ist. Durch folgendes Programm können Sie die Zeichen sichtbar machen :

```
30 PRINT A$,ASC(A$+CHR$(0))
```

Dadurch erreichen Sie, daß jedes Zeichen als Buchstabe und Asciiwert ausgedruckt wird. Trifft die GET#-Anweisung auf ein Nullbyte, wird es als Leerstring interpretiert. Damit bei der Umwandlung in den ASCII-Code dadurch keine Fehlermeldung ausgelöst wird, wird zu A$ immer CHR$(0) addiert.

Spulen Sie nun zurück und starten das Programm erneut. Auf Ihrem Bildschirm erscheint nun folgendes:

```
S         83
T         84
R         82
I         76
N         78
G         71
1         49
                    44
S         83
.
.
.
N         78
G         71
4         52
```

13
0
32

.
.
.

Die Null nach STRING4 ist die Kennzeichnung für das Dateiende. Danach folgt nur noch "SPACE".

3. DIE SEKUNDÄRADRESSE

Wie ich schon im zweiten Kapitel geschrieben habe, gibt es neben der Geräteadresse auch eine Sekundäradresse. Diese Adresse wird allgemein dazu benutzt, dem Peripheriegerät oder dem Computer als Empfänger eine weitere Anweisung zur Betriebsart zu übermitteln.

Vorweg möchte ich noch ein paar grundlegende Bemerkungen über die zwei verschiedenen Programmarten machen. Zum einem gibt es Basic-Programme. Diese Programme sind nicht abhängig davon, in welchem Speicherbereich sie stehen, sie sind verschiebbar.

Bei dem VC 20 zum Beispiel hängt die Lage des Basic-Speicher- Bereiches davon ab, wie er erweitert ist. In der Grundversion startet der Basic-Bereich bei der Speicherstelle 1025. Erweitern Sie ihn z.B. um 8 kByte, beginnt der Basic-Bereich bei 1609. Trotzdem laufen Programme, die auf der Grundversion geschrieben worden sind, auch auf der erweiterten Version.

Damit diese Programme auch immer an den Anfang des Basic-Bereiches geladen werden, besitzt der VC 20 und auch der C-64 einen sog. Relativlader, der automatisch alle Basic-Programme an den Basic-Start einliest.

Die andere Programmart ist das Maschinenprogramm. Diese Art von Programme laufen nur in dem Speicherbereich, für den sie geschrieben wurden. Sie dürfen nicht relativ geladen werden.

Um Basic-Programme von Maschinenprogrammen unterscheiden zu können, wird die Sekundäradresse bei der Cassettenhandhabung benutzt.

Im einzelnen bewirken die Sekundäradressen folgendes:

I. SAVE

0 - Das abgespeicherte Programm wird als Basic-Programm gekennzeichnet. Dadurch wird erreicht, daß dieses Programm mit dem Befehl LOAD automatisch an den gerade aktuellen Basic-Start, also relativ, geladen wird.

1 - Sie kennzeichnet ein Programm als Maschinenprogramm. Wenn Sie ein so gespeichertes Programm mit LOAD laden, wird es automatisch an die Adresse geladen, von welcher es abgespeichert wurde.

2 - Sie speichert ein Programm als Basic-Programm ab wie mit der Sekundäradresse 0. Zusätzlich schreibt der Rechner hinter das Programm noch einen sog. EOT (End Of Tape) Block. Dieser Block sagt dem Rechner beim Lesen, daß er hier mit der Suche nach weiteren Programmen aufhören soll. Trifft der Computer beim Suchen eines Programms auf solch einen Block, ohne vorher ein Programm gefunden zu haben, so gibt er die Meldung aus "?FILE NOT FOUND ERROR" aus.

3 - Sie kennzeichnet ein Programm als Maschinenprogramm und schreibt einen EOT Block dahinter.

Bei dem Befehl LOAD hat die Sekundäradresse folgende Bedeutung:

II. LOAD

0 - Das Programm wird entsprechend der Information geladen, die im Programmkopf steht. D.h. als Maschinenprogramme gespeicherte Programme werden auch als solche geladen. Das gleiche gilt für Basic-Programme, sie werden relativ geladen.

1 - Jedes Programm, egal wie es gekennzeichnet wurde, wird absolut, also an die Adresse geladen, von welcher es abgespeichert wurde.

Beim OPEN Befehl gibt es folgende Sekundäradressen:

III. OPEN

0 - Sie öffnet eine Datei zum Lesen.

1 - Sie öffnet eine Datei zum Schreiben.

2 - Sie öffnet eine Datei zum Schreiben und schreibt einen EOT-Block hinter die Datei.

4. DIE STATUSVARIABLE

Wie ich schon in den vorausgegangenen Kapiteln erwähnt habe, gibt es in Ihrem Commodore eine festdefinierte Variable, die Ihnen Auskunft über den Verlauf einer Cassettenoperation gibt. Diese sog. Statusvariable ST wird vom Computer bei jedem Zugriff auf Cassette oder andere Peripherigeräte gesetzt. Mit ihr haben Sie die Möglichkeit, eventuell aufgetretene Fehler genauer zu spezifizieren. Weiterhin können Sie mit ihr feststellen, wann eine Datei zu Ende ist.

Diese Variable besteht aus acht einzelnen "Flags", die entsprechend den aufgetretenen Fehlern gesetzt werden. Mit Hilfe einer AND Verknüpfung können Sie jedes einzelne Bit dieser Variable testen. Näheres über die verschiedenen Zahlendarstellungen und logischen Verknüpfungen finden Sie in dem Buch VC 20 Tips & Tricks.

Aus der Tabelle 1 können Sie die genaue Bedeutung der einzelnen Flags entnehmen.

Um das vierte Bit der Statusvariablen zu verstehen, muß man wissen, daß der Computer alle Daten und Programme zweimal auf Band schreibt und beim Einlesen beide Versionen miteinander vergleicht. Näheres finden Sie dazu noch im Kapitel 8.

Tabelle 1

ST-Bit	ST-Dez.-Äquiv.	Bedeutung
0	1	Keine Bedeutung für die Cassette
1	2	- " -
2	4	Kurzer Block. Ein gefundener Block ist kürzer, als er sein müßte.
3	8	Langer Block. Ein gefundener Block ist länger, als er sein müßte.
4	16	Second Pass Fehler. Die Daten des ersten Pass stimmen nicht mit denen des zweiten überein.
5	32	Prüfsummenfehler. Die abgespeicherte Prüfsumme stimmt nicht mit der errechneten überein.
6	64	Fileende
7	128	Bandende. EOT wurde gelesen.

Wie können Sie jetzt diese Variable in Ihren Programmen verwenden ? Wie Sie aus der Tabelle entnehmen können, gibt das 6. Bit darüber Auskunft, ob eine Datei zu Ende ist. Wenn Sie also innerhalb Ihres Programms Bit sechs abfragen, können Sie das Einlesen nach dem letzten String beenden. Geben Sie einmal folgendes Programm ein und lesen Sie damit die im Kapitel 2 angelegte Datei ein. Nachdem "STRING4" ausgedruckt wurde, erscheint READY und der Cursor.

```
10 OPEN1,1,0,"TEST"
20 INPUT#1,A$
30 PRINT A$
40 IF(ST AND 64) = 64 THEN60
50 GOTO20
60 CLOSE1
70 END
```

4.1 PROGRAMME RETTEN NACH LOAD ERROR

Ihnen ist es bestimmt schon passiert, daß sich Ihr Computer nach dem Ladevorgang mit "?LOAD ERROR" meldete und das geladene Programm gar nicht oder nur teilweise eingelesen wurde.

Die Ursache dafür ist meistens, daß die Position des Tonkopfes relativ zum Band beim Schreiben eine andere war als beim Lesen. Abhilfe schafft da nur eine neue Justierung des Tonkopfes, wie sie im Kapitel 11.5 beschrieben wird.

Läßt sich das Programm nach neuer Tonkopfeinstellung immer noch nicht laden, liegt der Fehler am Band oder an einer fehlerhaften Speicherung des Programms. Sie brauchen aber nicht zu verzweifeln. Unter bestimmten Bedingungen ist es möglich, das Programm zumindest teilweise zu retten.

Wenn Sie mit Ihrem Computer ein Programm abspeichern, wird es zweimal hintereinander auf Band geschrieben. Beim Einlesen wird die erste Version in den Speicher geladen und dann mit der zweiten verglichen. Stellt der Rechner dabei einen Fehler fest, versucht er erst diesen Fehler zu korrigieren. Ist das nicht möglich, setzt er Bit vier in der Statusvariablen und gibt ein "?LOAD ERROR" aus.

Es kann auch sein, daß er einzelne Bits oder Bytekennzeichnung nicht lesen kann, was die Synchronisation zwischen Leseroutine und eingelesener Information "außer Tritt" bringt. Das hat zur Folge, daß die Bits zwei und/ oder drei gesetzt werden.

Jeder aufgetretene Fehler führt dazu, daß die Basic-Vektoren 45, 46 (Programmende) nicht entsprechend gesetzt werden.

Falls der Fehler erst gegen Ende des Programms oder erst im zweiten Pass aufgetreten ist, können Sie das Programm ganz oder zumindest teilweilse listen. Diesen listbaren Teil können Sie mit einem UNNEW-Befehl retten.

Dies ist ein Befehl, der nicht im Commodore Basic 2.0 enthalten ist. Besitzen Sie keine Basic-Erweiterung, die diesen Befehl enthält, können Sie das am Ende dieses Kapitels abgedruckte Programm eingeben. Falls Sie kein MONITOR-Programm haben, geben Sie es mit dem Basic-Lader ein und speichern es ab. Danach starten Sie den Lader. Nun können Sie das Maschinenprogramm von der angezeigten Startadresse bis zum Basic-RAM-Ende abspeichern (siehe dazu Kapitel 6). Das so abgespeicherte Maschinenprogramm können Sie nun bei Bedarf einfach mit LOAD zuladen und mit SYS(Startadresse) starten, um entweder ein gelöschtes Programm wieder ins Leben zu rufen, oder um ein nur fehlerhaft ladbares Programm zumindest teilweise zu retten.

Nachdem Sie mit dem UNNEW-Befehl die Programmendvektoren gesetzt haben, speichern Sie es auf einer anderen Cassette ab. Nun kommt der schwierigste Teil. Jetzt müssen Sie feststellen, wo das Programm beschädigt ist. Das geschieht auf die gleiche Weise, wie Sie ein anderes Basic-Programm auf Fehler untersuchen.

Wie schon anfangs bemerkt, ist der häufigste Grund für einen "?LOAD ERROR" ein dejustierter Kopf. Darum sollten Sie als erstes immer versuchen, den Tonkopf besser auf das Band zu justieren und ggf. die oben beschriebene Prozedur mit verschiedenen Tonkopfeinstellungen zu wiederholen, bis Sie ein möglichst wenig beschädigtes Programm laden können.

```
************************************************************
  100 REM UNNEW 64
************************************************************
  110 PS=0:E=256*PEEK(56)+PEEK(55)-1:A=E-51
  120 FOR I=A TO E:READ X:PS=PS+X:POKE I,X:NEXT
  130 IF PS<>5274 THEN PRINT"DATA ERROR":END
  140 H=INT(A/256):POKE 56,H:POKE 55,A-256*H
  150 PRINT"STARTADRESSE"A:NEW
  160 DATA 165,43,164,44,133,34,132,35,160,3,200,177,34,20
      8,251,200,152,24,101
  170 DATA 34,160,0,145,43,165,35,105,0,200,145,43,32,51,1
      65,165,34,105,2,133
  180 DATA 45,165,35,105,0,133,46,32,99,166,76,123,227
```

```
************************************************************
  100 REM UNNEW 20
************************************************************
  110 PS=0:E=256*PEEK(56)+PEEK(55)-1:A=E-51
  120 FOR I=A TO E:READ X:PS=PS+X:POKE I,X:NEXT
  130 IF PS<>5319 THEN PRINT"DATA ERROR":END
  140 H=INT(A/256):POKE 56,H:POKE 55,A-256*H
  150 PRINT"STARTADRESSE"A:NEW
  160 DATA 165,43,164,44,133,34,132,35,160,3,200,177,34,20
      8,251,200,152,24,101
  170 DATA 34,160,0,145,43,165,35,105,0,200,145,43,32,51,1
      97,165,34,105,2,133
  180 DATA 45,165,35,105,0,133,46,32,99,198,76,103,228
```

```
120:                         ;
130:                         ; UNNEW-BEFEHL
140:                         ;
150:                         ; AUFRUF MIT SYS STARTADRESSE
160:                         ;
170:                         ;
172:    7000                          .OPT P1,00
180:    7000                          .PAG 61
182:    A663                 CLR      =    $A663    ;($C663)
184:    E37B                 BASIC    =    $E37B    ;($E467)
186:    A533                 BINDEN   =    $A533    ;($C533)
200:    7000                          *=   $7000
210:    7000 A5 2B                    LDA  $2B      ;BASICSTART IN PUFFER
220:    7002 A4 2C                    LDY  $2C
230:    7004 85 22                    STA  $22      ;SCHREIBEN
240:    7006 84 23                    STY  $23
250:    7008 A0 03                    LDY  #3       ;NULL-BYTE SUCHEN
260:    700A C8              NULL     INY
270:    700B B1 22                    LDA  ($22),Y
280:    700D D0 FB                    BNE  NULL
290:    700F C8                       INY           ;ZEILENLAENGE
292:    7010 98                       TYA
294:    7011 18                       CLC
300:    7012 65 22                    ADC  $22      ; + BASICSTART ERGIBT KOPPEL-
310:    7014 A0 00                    LDY  #0       ;ADRESSE DER ERSTEN ZEILE
320:    7016 91 2B                    STA  ($2B),Y   ;KOPPPELADR AN BASICSTART
330:    7018 A5 23                    LDA  $22 + 1   ;SCHREIBEN
340:    701A 69 00                    ADC  #0
350:    701C C8                       INY
360:    701D 91 2B                    STA  ($2B),Y
370:    701F 20 33 A5                 JSR  BINDEN   ;KOPPELADRESSEN NEU BERECHNEN
460:    7022 A5 22                    LDA  $22      ;IN $22/$23 STEHT ENDADRESSE
470:    7024 69 02                    ADC  #2       ; + 2 ERGIBT BASICENDVECTOR
480:    7026 85 2D                    STA  $2D
490:    7028 A5 23                    LDA  $22 + 1
500:    702A 69 00                    ADC  #0
510:    702C 85 2E                    STA  $2E
520:    702E 20 63 A6                 JSR  CLR      ;SETZT UEBRIGE ZEIGER
530:    7031 4C 7B E3                 JMP  BASIC    ;RUECKKEHR ZUM BASIC
ü7000-7034
```

5. LADEN UND SPEICHERN VOM PROGRAMM AUS

Bei der Behandlung der Befehle SAVE und LOAD im Kapitel 2 habe ich beschrieben, wie man Programme im Direktmodus abspeichert und lädt. Diese Befehle können aber auch innerhalb von Programmen stehen.

Der SAVE-Befehl arbeitet innerhalb von Programmen genauso wie im Direktmodus. Es ist somit möglich, ein Programm sich selber abspeichernzulassen. Diese Möglichkeit werden Sie wahrscheinlich aber nur selten nutzen. Weit häufiger möchte man nur einen Teil eines Basic-Programms oder ein Maschinenprogramm abspeichern.

Stellen Sie sich vor, Sie haben ein Programm geschrieben, das ab Zeile 10000 Datazeilen generiert. Diesen Programmteil wollen Sie alleine abspeichern, um ihn dann an andere Programme anhängen zu können. Sie müssen als erstes feststellen, ab welcher Speicherstelle die Zeile 10000 im Speicher steht, da Sie ja nur ab Zeile 10000 abspeichern wollen. Mit den folgenden Programmzeilen lösen Sie dieses Problem:

```
1000 S = PEEK(43) + 256 * PEEK(44) : BS = S
1010 ZN = PEEK(S + 2) + 256 * PEEK(S + 3)
1020 IF ZN = 10000 THEN 1050
1030 S = PEEK(S) + 256 * PEEK(S + 1)
1040 GOTO 1010
```

Um diese Programmzeilen zu verstehen, muß ich Ihnen kurz erklären, wie Basic-Zeilen im Speicher stehen.

Die ersten beiden Bytes einer Basic-Zeile bildet die sogenannte Koppeladresse. Sie gibt die Startadresse der nächsten Basic-Zeile an, zeigt also auf die nächste Koppeladresse. In den beiden folgenden Bytes ist die Zeilennummer abgespeichert. Danach kommt der Inhalt der

Basic-Zeile, abgeschlossen durch ein Nullbyte. Ihr Basic-Programm steht also folgendermaßen im Speicher :

KAL KAH ZNL ZNH Basictext 00 KAL KAH ZNL ZNH ....

Hierbei bedeuten :

KAL Koppeladresse LSB
KAH Koppeladresse MSB
ZNL Zeilennummer LSB
ZNH Zeilennummer MSB

Bei unserem kleinen Programm wird in der Zeile 1000 die Programmstartadresse aus den Vektoren in S und BS eingelesen. In der Zeile 1010 wird die Zeilennummer in die Variable ZN übertragen. Die Abfrage in der Zeile 1020 testet, ob die gewünschte Zeilennummer erreicht wurde. Wenn ja, wird in das weitere Programm verzweigt. Ist sie noch nicht erreicht, wird in Zeile 1030 aus der Koppeladresse die Startposition der nächsten Basic-Zeile in S geschrieben und über Zeile 1040 wieder nach 1010 gesprungen.

Nach diesem Programm haben Sie alle Informationen, um den letzten Teil Ihres Programms abzuspeichern. Wichtig ist nun nur noch, daß Sie nach dem Speichervorgang wieder den alten Wert in den Basicstartvektor schreiben. Mit dem folgenden Programmteil können Sie die Datazeilen abspeichern.

```
1050 POKE 43,S AND 255: POKE 44,INT(S/256) :REM ZEIGER
     AUF ZEILE 10000 SETZEN
1060 SAVE"DATAZEILEN" :REM DATAZEILEN SPEICHERN
1070 POKE 43,BS AND 255: POKE 44,INT(BS/256) :REM ZEIGER
     ZURUECKSETZEN
```

Wenn Sie Maschinenprogramme abspeichern wollen, wird das Ganze etwas komplizierter. Lesen Sie dazu bitte Kapitel 6.

5.1 OVERLAYTECHNIK

Der LOAD-Befehl arbeitet innerhalb von Programmen etwas anders als im Direktmodus. Im Kapitel 2 habe ich geschrieben, daß nach einem LOAD die Basic-Zeiger auf das neue Programm eingestellt werden. Geben Sie aber die LOAD-Anweisung innnerhalb von Programmen, bleiben die Vektoren erhalten, ebenso wie alle bis dahin definierten Variablen.

Daher ist es möglich, von einem Hauptprogramm aus verschiedene Unterprogramme aufzurufen, die alle mit den gleichen Variablen arbeiten. Diese Methode nennt man OVERLAYTECHNIK.

Bei dieser Technik sind aber einige Dinge zu beachten:

1. Da die Basic-Vektoren nicht verändert werden, muß das aufrufende Programm mindestens so lang sein wie das aufgerufene.

   Direkt hinter einem Basic-Programm werden die Variablen abgespeichert, und der Vektor 45, 46 ist damit auch der Zeiger für den Beginn der Variablentabelle. Ist nun das nachgeladene Programm länger, werden die Variablen überschrieben, und der Zeiger weist in das neue Programm. Sobald nun eine Variablenoperation stattfindet, wird die Variable innerhalb des Programms gesucht, was zum "Absturz" des Rechners führen kann.

   Mit einem Trick ist es aber möglich, von einem kurzen Programm aus ein längeres aufzurufen.

   Zuerst laden Sie das längste nachzuladende Programm im Direktmodus ein und stellen durch Auslesen der Speicherstellen 45,46 seine Länge fest. Die erhaltenen Werte notieren Sie sich am besten. Als nächstes laden Sie das aufrufende Programm und setzen folgende Basic-Zeile an den Anfang:

```
10 POKE 45,W1: POKE 46,W2: CLR
```

Die Zeilennummer ist wahlweise, sie muß nur die kleinste in Ihrem Programm sein. W1 und W2 stehen für die von Ihnen notierten Werte.

Durch die POKE-Befehle verlängern Sie Ihr Programm künstlich. Das CLR bewirkt, daß auch die anderen Zeiger entsprechend geändert werden.

2. Sie müssen beachten, daß nach einem LOAD das Programm wieder von Anfang an ausgeführt wird.

   Das ist nur dann sinnvoll und unproblematisch, wenn Sie Basic-Programme nachladen. Wollen Sie aber ein Maschinenprogramm nachladen, ändert sich das im Speicher befindliche Basic-Programm nicht und startet sich immer wieder von neuem. Wenn das nicht geschehen soll, müssen Sie das durch eine Sprungtabelle verhindern. Bei dem folgenden Programm werden am Anfang des Programms drei Maschinenprogramme eingeladen, ehe es zum Hauptprogramm übergeht.

```
10 REM START
20 IF A = 0 THEN A=1: LOAD"MPR1",1,1
30 IF A = 1 THEN A=2: LOAD"MPR2",1,1
40 IF A = 2 THEN A=3: LOAD"MPR3",1,1
50 REM HAUPTPROGRAMM
```

3. Als letztes ist eine Besonderheit der Stringspeicherung zu beachten.

   Ihr Commodore Betriebssystem ist so konzipiert, daß möglichst wenig Speicherplatz für Strings verwendet wird. Jedesmal, wenn Sie einer Stringvariablen einen String zuweisen, wird in der Variablentabelle unter dem Variablennamen ein Zeiger abgelegt, der auf den betreffenden String zeigt. Weisen Sie einer Variablen durch eine INPUT - Anweisung einen String zu, wird dieser

in einer speziellen Tabelle am Ende des Basic-RAMs abgelegt. Geschieht die Zuweisung aber innerhalb des Programms, z.B. durch

```
100 A$="STRING"
```

wird der String nicht noch einmal in die Stringtabelle übertragen, sondern der Zeiger wird auf die Position des Strings innerhalb des Programms gesetzt.

Laden Sie nun ein Programm nach, steht an der entsprechenden Stelle etwas ganz anderes und der Computer druckt bei dem Befehl PRINT A$ nur wirres Zeug aus. Abhilfe schaffen Sie sich dadurch, daß Sie scheinbar einen neuen String definieren, indem Sie den im Programm definierten mit einem Leerstring verknüpfen. Schreiben Sie also:

```
100 A$="STRING": A$=A$+""
```

Dadurch bringen Sie das Betriebssystem dazu, den String in die Stringtabelle zu übertragen.

6. ABSPEICHERN VON MASCHINENPROGRAMMEN

Wenn Sie sich etwas länger mit Ihrem Commodore beschäftigt haben, stehen Sie irgendwann vor dem Problem, bestimmte Speicherbereiche abspeichern zu wollen. Diese bestimmten Speicherbereiche können z.B. der Bildschirminhalt oder ein Maschinenprogramm sein. Leider unterstützt das Commodore Basic diesen Wunsch überhaupt nicht. Daß es mit einigen Tricks doch geht, haben Sie schon im Kapitel 5 gelesen.

In den letzten Kapiteln habe ich geschrieben, daß der Rechner bei einem SAVE-Befehl immer den Bereich, der zwischen den Zeigern in 43, 44 und 45, 46 liegt, abspeichert. Was liegt da näher, als die Zeiger so zu ändern, daß sie auf den Start und das Ende des von Ihnen gewünschten Speicherbereichs zeigen. Wenn Sie aber nach der Speicherung noch 'normal' mit Ihrem Computer arbeiten wollen, dürfen Sie es sich nicht so leicht machen.

Damit der Rechner nach der Speicherung so arbeitet wie vorher, muß in den Vektoren wieder der alte Wert stehen. Sie müssen diese Werte also vorher 'retten' und nachher wieder in die Zeiger 43-46 übertragen.

Um die Inhalte der Zeiger zu retten, dürfen Sie sie nicht Variablen übergeben, da, wie schon in den letzten Kapiteln beschrieben, der Programmendvektor gleichzeitig der Variablenstartvektor ist. Sobald Sie diesen Zeiger verändern, findet der Computer keine Variablen mehr. Aus diesem Grund müssen Sie die Zeigerinhalte in Speicherstellen POKEn, die nicht durch den Speichervorgang oder das Betriebssystem verändert werden.

Im folgenden Beispiel verwende ich die Speicherstellen 251 bis 254. Diese Stellen werden vom Betriebssystem nicht benutzt. Mit diesen Zeilen schreiben Sie den Bildschirminhalt auf Cassette :

```
10 POKE251, PEEK(43): POKE252, PEEK(44): REM START RETTEN
20 POKE253, PEEK(45): POKE254, PEEK(46): REM ENDE  RETTEN
30 POKE43,0: POKE 44,4: REM VIDEORAMSTART ALS START
40 POKE45,232: POKE 46,3: REM VIDEORAMENDE ALS ENDE
50 SAVE"BILDSCHIRM",1,1 : REM VIDEORAM SPEICHERN
60 POKE43, PEEK(251): POKE44, PEEK(252):REM START ZURUECK
70 POKE45, PEEK(253): POKE46, PEEK(254):REM ENDE  ZURUECK
```

Sie sehen, daß das Ganze recht kompliziert ist. Mit einem kleinen Maschinenprogramm geht es viel einfacher. Sie geben einfach

```
SYS adr"NAME",GA,sadr,eadr
```

mit adr : Startadresse des Maschinenprogramms
NAME : Name des abzuspeichernden Bereichs
GA : Geräteadresse (Cassette = 1)
sadr : Startadresse des abzuspeichernden Bereichs
eadr : Endadresse - ´´ -

ein, und der Bereich wird abgespeichert, ohne daß Sie irgendwelche Manipulationen mit den Vektoren vornehmen müssen.

Die abgedruckten Basic-Lader speichern das kleine Maschinenprogramm an das Ende des Basic-RAMs und geben die Startadresse aus, mit welcher Sie es aufrufen können. Bevor Sie aber den Lader starten, speichern Sie Ihn ab, weil er sich automatisch löscht.

Für Interessierte möchte ich hier einige Erklärungen zum Maschinenprogramm geben.

In Zeile 310 wird eine Betriebssystemroutine aufgerufen, die den Namen vom Bildschirm liest und ihn in die entsprechenden Speicherstellen schreibt. Die Routine, die in Zeile 320 aufgerufen wird, prüft, ob das nächste Zeichen ein Komma ist, und überträgt den folgenden Ausdruck in das X-Register.

In Zeile 330 wird die Sekundäradresse in das Y-Register und in Zeile 340 die logische Filenummer in den Akku geladen. Mit diesen Parametern wird eine Routine aufgerufen, die die Fileparameter setzt. Diese Routine muß vor jedem Lade- und Speichervorgang aufgerufen werden.

In den Zeilen 350 - 380 wird wiederum auf ein Komma getestet, um dann die Startadresse einzulesen und zu verarbeiten. Sie wird in den Speicherstellen $14 und $15 abgelegt. In den folgenden Zeilen werden die Inhalte von $14/$15 nach $FB/$FC gerettet, um dann wie davor, die Endadresse einzulesen.

Die Übergabe dieser Parameter an die SAVE-Routine erfolgt dadurch, daß in den X- und Y-Registern die Endadressen absolut übergeben werden, und der Akku die Zeropage-Position enthält, in der die Startadresse steht.

```
100 REM SAVE ADRESSE 64
110 E=256*PEEK(56)+PEEK(55)-1:A=E-47
120 FOR I=A TO E:READ X:POKE I,X:NEXT
130 H=INT(A/256):POKE 56,H:POKE 55,A-256*H
140 PRINT"STARTADRESSE"A:NEW
150 DATA 32,87,226,32,0,226,160,1,169,0,32,186,255,32,14
    ,226,32,138,173,32,247
160 DATA 183,166,20,164,21,134,251,132,252,32,14,226,32,
    138,173,32,247,183,169
170 DATA 251,166,20,164,21,76,216,255
```

```
100 REM SAVE ADRESSE VC 20
110 E=256*PEEK(56)+PEEK(55)-1:A=E-47
120 FOR I=A TO E:READ X:POKE I,X:NEXT
130 H=INT(A/256):POKE 56,H:POKE 55,A-256*H
140 PRINT"STARTADRESSE"A:NEW
150 DATA 32,84,226,32,253,225,160,1,169,0,32,186,255,32,
    11,226,32,138,205,32
160 DATA 247,215,166,20,164,21,134,251,132,252,32,11,226,
    32,138,205,32,247,215
170 DATA 169,251,166,20,164,21,76,216,255
```

```
80:     C000                    .PAG 61
100:    C000                    .OPT P1,O
102:    C000
102:    C000
103:                 ;                SAVE ADRESSE
104:                 ;
105:                 ;                 AUFRUF MIT
107:                 ;
110:                 ; SYS ADRESSE "NAME",GA,STARTADR.,ENDADR.
111:                 ;
112:    C000
112:    C000
114:                 ;    WERTE IN KLAMMERN GELTEN FUER
117:    C000
117:    C000
119:                 ;                  VC 20
120:    C000
120:    C000
125:    C000                    *=   $C000
130:    E257         SETFNAME =      $E257   ;($E254)
140:    E200         HBYTE    =      $E200   ;($E1FD) KOMMATEST U. BYTE => X
150:    FFBA         SETLFS   =      $FFBA
160:    E20E         CHRIN    =      $E20E   ;($E20B) KOMMATEST
170:    AD8A         FRMNUM   =      $AD8A   ;($CD8A)
180:    B7F7         FACINT   =      $B7F7   ;($D7F7) FAC NACH $14,$15
190:    FFD8         SAVE     =      $FFD8
200:    00FB         PUFFER   =      $FB
300:    C000
300:    C000
310:    C000 20 57 E2           JSR  SETFNAME  ; FILENAME HOLEN
320:    C003 20 00 E2           JSR  HBYTE   ; GA HOHLEN
330:    C006 A0 01              LDY  #1      ; SA
340:    C008 A9 00              LDA  #0      ; LFN
350:    C00A 20 BA FF           JSR  SETLFS  ;FILEPARAMETER SETZEN
360:    C00D 20 0E E2           JSR  CHRIN   ;STARTADRESSE HOLEN
370:    C010 20 8A AD           JSR  FRMNUM  ;UND AUSWERTEN
380:    C013 20 F7 B7           JSR  FACINT
390:    C016 A6 14              LDX  $14
400:    C018 A4 15              LDY  $15
410:    C01A 86 FB              STX  PUFFER  ;IN PUFFER RETTEN
420:    C01C 84 FC              STY  PUFFER+1
430:    C01E 20 0E E2           JSR  CHRIN   ;ENDADRESSE HOLEN
440:    C021 20 8A AD           JSR  FRMNUM  ;UND AUSWERTEN
450:    C024 20 F7 B7           JSR  FACINT
460:    C027 A9 FB              LDA  #PUFFER  ;UEBERGABE STARTADR.
470:    C029 A6 14              LDX  $14     ;UEBERGABE ENDADR.
480:    C02B A4 15              LDY  $15
490:    C02D 4C D8 FF           JMP  SAVE
üC000-C030
```

7. DER CASSETTENPUFFER

In diesem Kapitel möchte ich näher auf den Cassettenpuffer eingehen. Dieser Puffer belegt den Speicherbereich von 828 (Hexadezimal 033c) bis 1019 (Hexadezimal 03FB). Dieser Puffer hat 2 grundsätzliche Aufgaben. Zum einen wird in diesem Bereich der Filekopf, der sogenannte Header, generiert und vor jedem File, sei es ein Programm- oder Datenfile, abgespeichert und dient dann zur Kennzeichnung desselben. Er enthält dann folgende Parameter:

1. Filetyp,	Byte 0	= 828
2. Startadresse,	Byte 1, 2	= 829, 830
3. Endadresse,	Byte 3, 4	= 831, 832
4. Filenamen ab	Byte 5	= 833

Die restlichen Bytes sind mit Space beschrieben.

Als zweite Funktion dient der Cassettenpuffer als Zwischenspeicher bei der Datensicherung. Die abzuspeichernden Daten werden zuerst in diesen Speicherbereich geschrieben und erst, wenn der ganze Puffer beschrieben wurde, wird er auf Band übertragen. Wenn Sie Daten laden, werden diese blockweise geladen und dann durch INPUT# oder GET# aus dem Puffer gelesen, wie auch im Kapitel 2.4 ff beschrieben.

Der Filetyp ist folgendermaßen codiert:

1 = Basicprogramm, relativ laden
2 = Datenblock
3 = Maschinenprogramm, absolut laden
4 = Daten Header
5 = EOT-Block

Im Header sind alle wichtigen Daten enthalten, die zur Kennzeichnung benötigt werden. Was liegt da näher, als mit

Hilfe dieser Daten, die mit Programmen und Daten bespielten Cassetten, zu archivieren.

7.1 ANLEGEN EINES CASSETTENINHALTSVERZEICHNISSES

Das Programm am Ende von 7.1 druckt ein Inhaltsverzeichnis einer Cassette aus. Falls Sie einen Drucker haben, können Sie sich das Inhaltsverzeichnis auch ausdrucken lassen.

PROGRAMMBESCHREIBUNG

Die Zeilen 100 - 160 dienen zur Initialisierung. Hier werden die einzelnen Tabulatorstops in Tx$ und die verschiedenen Filetypen in das Feld TN$ eingelesen.

Für den Fall, daß die Ausgabe auch zu einem Drucker geschickt werden soll, wird in Zeile 180 ein Druckerkanal eröffnet und in 190 - 220 die Überschrift an den Drucker übermittelt. In der Zeile 230 wird dann eine Datei zum Cassettenlesen ohne Namen eröffnet. Dadurch wird der nächste Header eingelesen.

Durch PEEK werden in den Zeilen 240 - 270 Typ-Byte, Start- und Endadresse und Filename aus dem Puffer gelesen und formatiert. In den Zeilen 280 - 290 wird aus Start- und Endadressen die Programmlänge in KByte berechnet.

Durch die Zeilen 300 - 320 werden die Daten auf Drucker und Bildschirm ausgegeben.

In Zeile 330 wird über die Programmlängen der Zählerstand berechnet. Nachdem dann die Datei geschlossen wurde, springt das Programm wieder zum OPEN - Befehl.

```
**********************************************************
 100 REM                    *INHALTSVERZEICHNIS***
**********************************************************
 110 PA=828:REM STARTADRESSE IM CASSETTENPUFFER
 120 PRINT"[DOWN2]DRUCKEN ? [J/N]  ";D$
 130 GET D$:IF D$=""THEN 130
 140 T1$=CHR$(16)+"05":T2$=CHR$(16)+"10":T3$=CHR$(16)+"20"
     :T4$=CHR$(16)+"30"
 150 T5$=CHR$(16)+"38":T6$=CHR$(16)+"50":T7$=CHR$(16)+"63"
     :REM TABULATORSTOPS
 160 TN$(0)="RELATIV":TN$(2)="ABSOLUT":TN$(3)="DATEI"
 170 IF D$<>"J"THEN 230
 180 OPEN 2,4
 190 PRINT#2,T1$"LFN"T2$"ZAEHLER"T3$"TYP"T4$"K-BYTE"T5$"
       ANFANG"T6$"   ENDE";
 200 PRINT#2,T7$"  NAME"
 220 PRINT#2
 230 OPEN 1,1
 240 TY=PEEK(PA):TY$=TN$(TY-1)
 250 A$=" :"+RIGHT$("      "+STR$(PEEK(PA+1)+(256*PEEK(PA
     +2))-1),6)
 260 B$=" -"+RIGHT$("      "+STR$(PEEK(PA+3)+(256*PEEK(PA
     +4))-1),6)
 270 C$=": ":FOR I=5 TO 20:C$=C$+CHR$(PEEK(PA+I)):NEXT
 280 T=VAL(RIGHT$(B$,6))-VAL(RIGHT$(A$,6))
 290 K$=RIGHT$("     "+STR$(INT(T/1024*100)/100),6)
 300 PRINT Z;TY$;K$;A$B$:PRINT C$:PRINT
 310 N=N+1:N$=RIGHT$("      "+STR$(N),3)
 315 IF D$<>"J"THEN 330
 320 PRINT#2,T1$;N$; T2$;Z; T3$;TY$; T4$;K$; T5$;A$; T6$;
     B$; T7$;C$
 330 Z=INT(Z+(T/212)+4+2**(N/100))
     :REM BANDZAEHLER BERECHNEN
 340 CLOSE 1
 350 GOTO 230
```

7.2 ANZEIGE DER GEFUNDENEN DATEIEN

Das Auslesen des Filenamens mittels eines PEEK-Befehls kann auch innerhalb eines Dateiverwaltungsprogramms nützlich sein. Haben Sie sich noch nicht darüber geärgert, daß, wenn Sie von einem Programm eine bestimmte Datei einlesen wollen, der Computer ewige Zeit sucht, um schließlich "?FILE NOT FOUND" auszugeben. Der Grund dafür war vielleicht, daß Sie das Band etwas zu weit vorgespult hatten.

Wenn der Rechner Ihnen aber sagt, welche Dateien er findet, können Sie leicht feststellen, ob Sie zu weit gespult haben oder nicht. Vorausgesetzt natürlich, daß Sie die ungefähre Reihenfolge der Dateien auf Band kennen.

Die folgenden Basic-Zeilen veranlassen den Computer, jede Datei, die er findet, anzuzeigen.

```
100 INPUT"FILENAME";F$
110 L=LEN(F$)
120 FE$=""
130 OPEN 4,1,0,FE$
140 FOR I=0 TO 15:FE$=FE$+CHR$(PEEK(833+I)):NEXT
150 PRINT:PRINT"FOUND ";FE$
160 IF LEFT$(FE$,L)<>LEFT$(F$,L)THEN CLOSE 4:GOSUB 200
    :GOTO 120
170 PRINT:PRINT"FILE WIRD GELADEN"
200 POKE 198,0:WAIT 198,1:RETURN
```

Durch den OPEN - Befehl ohne Namen in Zeile 130, veranlassen Sie den Rechner, jeden Datenkopf in den Puffer zu laden.

In Zeile 140 - 160 wird dann der Filename mittels PEEK aus dem Puffer in FE$ übertragen und in Zeile 170 angezeigt. Ergibt der Vergleich von FE$ mit dem gewünschten Filenamen in Zeile 180 keine Übereinstimmung, wird die Suche fortgesetzt. Anderenfalls wird die Datei geladen.

7.3 SELBSTSTARTENDE PROGRAMME UND PROGRAMMSCHUTZ

Bis jetzt haben wir nur Daten aus dem Cassettenpuffer ausgelesen. Dieser Puffer eignet sich aber auch vortrefflich, um Daten oder Programme aufzunehmen. So ist es möglich, durch diesen Puffer noch weitere Daten oder ein Programm mit einem anderen Programm abzuspeichern.

Der Puffer belegt den Speicherbereich von 828 bis 1019. Er ist also 191 Zeichen lang. Zur Informationsübergabe werden, wenn er als Header abgespeichert wird, maximal 16 + 2 + 2 + 1 (Filename, Startadresse, Endadresse, Typebyte) = 21 Byte benötigt. Es bleiben also noch 170 Byte übrig, ausreichend Platz, um kleine Maschinenprogramme unterzubringen.

Als Beispiel beschreibe ich Ihnen hier ein kleines Programm, das ein Vorprogramm erzeugt, das Ihren Wünschen entsprechend z.B. das nächste Programm automatisch lädt und startet.

Solche Vorprogramme können auch dazu dienen, Ihre Programme zu schützen. Ebenfalls kann mit solch einem Vorprogramm erreicht werden, daß vor dem Laden des Hauptprogramms der Basic-RAM verschoben wird, um vor das Basic-Programm noch eine oder mehrere Grafikseiten abzulegen. Dem Programmierer sind hier keine Grenzen gesetzt. Doch nun zu unserem Beispiel.

Ich habe dieses Beispiel gewählt, da es einerseits die allgemeine Technik zeigt, und andererseits auch von einem "Nur-Anwender" recht allgemein genutzt werden kann.

Hier ist die Programmbeschreibung:

In Zeile 10 wird der Filename abgefragt, unter welchem das Hauptprogramm abgespeichert werden soll. In Zeile 20 und 30 wird der Filename durch Auffüllen mit "SPACE" auf eine Länge von 16 Zeichen gebracht. Durch die Zeilen 40 bis 70 wird an den Filenamen ein Maschinenprogramm angehängt, welches in

den Datazeilen ab Zeile 180 steht.

Da die Programme für VC20 und C-64 bis auf die Datas identisch sind, sind von dem Lader-Programm für den C-64 nur die Datazeilen ausgedruckt.

Über die Zeilen 80 und 90 haben Sie nun die Möglichkeit, Basic-Befehle einzugeben, die später nach dem Laden des Vorprogramms automatisch ausgeführt werden.

Wenn Sie also wünschen, daß ein folgendes Basic-Programm geladen und gestartet werden soll, geben Sie LOAD RETURN RUN RETURN RETURN in Kurzbefehlen ein:

L┌↑R /↑↑

Aus programmtechnischen Gründen, die ich später beschreiben werde, ist es bei diesem Programm nicht möglich, mehr als 10 Zeichen zu übergeben. Diese Zeichen werden dann an den Filenamen hinter das Maschinenprogramm angehängt (Zeile 100 - 150).

In Zeile 160 wird der Zeiger zu einer Systemroutine, der Eingabe-Warteschleife, auf das Maschinenprogramm im Cassettenpuffer umgesetzt.

Die Zeile 170 speichert dann mt Hilfe des im letzten Kapitel beschriebenen Maschinenprogramms, diesen Vektor ab. Sie müssen also hinter den SYS-Befehl die Startadresse schreiben, die Ihr SAVE ADRESS-Programm hat.

Durch diesen Speicherbefehl wird der Filename mit Maschinenprogramm und Basic-Befehlen in den Puffer und auf Band geschrieben. Nach dem Programmende springt der Computer über den Vektor in 770, 771 normalerweise in die Eingabe-Warteschleife. Da dieser aber in Zeile 160 auf unser Maschinenprogramm im Cassettenpuffer gesetzt wurde, springt er dorthin und arbeitet es ab. Da wir im Ausgabestring dem Rechner den Befehl LOAD gegeben haben, versucht er das nächste Programm zu laden. Durch Drücken der

RUN/STOP - Taste kann man diesen Vorgang unterbrechen, um das Hauptprogramm hinter das Ladeprogramm zu speichern, das vom Vorprogramm geladen werden soll.

Wenn Sie ein Programm hinter den Lader geschrieben haben, spulen Sie das Band zurück zum Start des Ladeprogramms und geben LOAD und RETURN ein.

Ihr Computer findet nun das Ladeprogramm, lädt dieses, lädt sofort danach das folgende Programm ein und startet es.

Für Interessierte und Fortgeschrittene möchte ich nun diese Technik und das Programm genau erklären.

Die zwei Grundlagen dieser Technik sind folgende:

1. Übergabe eines Maschinenprogramms in den Cassettenpuffer durch Anhängen an den Filenamen,

2. Änderung von Vektoren, z.B. dem zur Eingabeschleife auf ein eigenes Maschinenprogramm und Speicherung eben dieser geänderten Vektoren. Im eigenen Maschinenprogramm muß dann vor dem Sprung in das geladene Programm der in das eigene Maschinenprogramm zeigende Vektor wieder auf den alten Wert gebracht werden.

3. Als Maschinenprogramme abgespeicherte Speicherbereiche werden immer an die Stelle geladen, von wo sie abgespeichert wurden.

Ihr Rechner besitzt in den Speicherstellen 768 - 819 eine Reihe von Vektoren, über welche er in bestimmte Unterprogramme springt. Durch Veränderung dieser Zeiger können Sie ein eigenes Programm dazwischenschalten.

Die Eingabe-Warteschleife ist das Unterprogramm, das Ihren Commodore geduldig auf irgendeine Eingabe warten läßt. Machen Sie eine Eingabe, wertet und führt er sie aus, um

anschließend wieder in diese Schleife zurückzukehren.

Durch unser Programm wird nun ein Vorprogramm erzeugt, das den Vektor zur Eingabe-Warteschleife mit der Startadresse eines eigenen Programms überschreibt, das mit dem Filenamen in den Speicher geschrieben wurde.

Wenn Sie selber weitere Experimente machen wollen, können Sie natürlich auch Ihr Maschinenprogramm z.B. in den Bereich $2C0 - $2FF (704 - 766) legen und als Vorprogramm den Bereich $2C0 - 303 abspeichern.

Doch sehen wir uns nun das Maschinenprogramm an. In den Zeilen 200 - 230 wird der Vektor zur Eingabe-Warteschleife wiederhergestellt. Ab der Speicherstelle $036C sind Ihre Eingaben abgespeichert. Diese werden in den Zeilen 250 bis 290 in den Tastaturpuffer übertragen. Durch den Sprung am Ende des Programms in die Eingabe-Warteschleife wird der Rechner veranlaßt, die Befehle, die im Tastaturpuffer stehen, auszuführen.

Das hier dargestellte Beispiel ist zwar recht einfach, hat aber den Nachteil, daß Sie nur wenige Befehle aufgrund des begrenzten Tastaturpuffers übergeben können. Sie können jetzt aber, wenn Sie das Prinzip verstanden haben, leicht dieses Programm auf Ihre eigenen Bedürfnisse hin abänderen, indem Sie im Maschinenprogramm ab Zeile 240 Ihre eigene Routine schreiben, und diese mit einem Sprung in die Eingabe - Warteschleife abschließen.

Ihnen ist bestimmt schon klar geworden, daß man diese Technik auch sehr gut zum Programm-Schutz verwenden kann, da sich das Programm ja selbst startet. Wenn Sie zusätzlich noch einen Programmabbruch durch die RUN/STOP - Taste unterbinden und das Hauptprogramm so abändern, daß es nur in Verbindung mit dem Lader lauffähig ist, hat ein fremder Anwender keine Möglichkeit, Einsicht in Ihr Programm zu nehmen.

Der ganze Schutz wäre aber nutzlos, wenn ich hier eine

genaue Anleitung geben würde. Ich hoffe aber, daß Sie nach meinen Ausführungen und mit etwas Maschinenprogrammierkenntnis selbst in der Lage sind, eigene Autostartroutinen zu entwickeln.

```
PROFI-ASS 64 V2.0     SEITE 1

110:   0351                  .OPT P1,O
120:   0351                  *=   $0351
130:   0351
130:   0351
140:             ;LADER PROGRAMM
150:   0351
150:   0351
160:   A483      VECTOR    =     $A483    ; ($C483) VECTOR
165:             ;                          IN DIE EINGABESCHLEIFE
170:   00C6      PANZ      =     $C6
180:   0277      PUFFER    =     $277     ; TASTATURPUFFER
185:   036C      TABEL     =     $36C     ; TABELLE DER BASIC-BEFEHLE
190:   0351
190:   0351
200:   0351 A9 83            LDA  #<VECTOR  ; SETZEN DES ALTEN
210:   0353 8D 02 03         STA  $302     ; VECTORS
220:   0356 A9 A4            LDA  #>VECTOR
230:   0358 8D 03 03         STA  $303
234:             ;
235:             ;  WERTE AUS TABELLE
237:             ;IN DEN TASTATURPUFFER
238:             ;    UEBERTRAGEN
239:             ;
240:   035B AE 6C 03         LDX  TABEL    ; ANZAHL DER ZEICHEN
250:   035E 86 C6            STX  PANZ
260:   0360 BD 6C 03 LOOP    LDA  TABEL,X
270:   0363 9D 77 02         STA  PUFFER,X
280:   0366 CA               DEX
290:   0367 D0 F7            BNE  LOOP
294:             ;
295:             ;IN DIE EINGABE-WARTESCHLEIFE
297:             ;    ZURUECKSPRINGEN
299:             ;
300:   0369 4C 83 A4         JMP  VECTOR
ü0351-036C
```

```
*************************************************************
    5 REM LADER FUER VC-20
*************************************************************
   10 INPUT"FILENAME ";F$
   20 SP$="                "
   30 F$=LEFT$(F$+SP$,16):S=833
   40 FOR I=0 TO 26
   50   :  READ F
   60    F$=F$+CHR$(F)
   70    NEXT
   80 PRINT"AUSGABESTRING MAX. 10 ZEICHEN "
   90 PRINT"↑ = RETURN, AM ENDE 2 MAL"
   95 INPUT A$
  100 LA=LEN(A$):IF LA>10 THEN 90
  110 F$=F$+CHR$(LA)
  120 FOR I=1 TO LA
  130   :  W$=MID$(A$,I,1):IF W$="↑"THEN W$=CHR$(13)
  140   :  F$=F$+W$
  150    NEXT
  160 POKE 770,81:POKE 771,3
  170 SYS 24528F$,1,768,772
  180 DATA 169,131,141,2,3,169,196,141,3,3,174,108,3,134,1
      98,189,108,3,157,119
  190 DATA 2,202,208,247,76,196,131


*************************************************************
    5 REM LADER FUER C-64
*************************************************************
   .
   .
   .

  180 DATA 169,131,141,2,3,169,164,141,3,3,174,108,3,134,1
      98,189,108,3,157,119
  190 DATA 2,202,208,247,76,131,164
```

8. SPEICHERFORMAT DER CASSETTENSPEICHERUNG

Um Daten oder Programme auf Band aufzuzeichnen, werden einzelne Bytes und Bits als Pulse aufgezeichnet.

Zur Codierung sind drei verschiedene Zeiten definiert:

K = kurz (176 Microsekunden)
M = mittel (256 Microsekunden)
L = lang (336 Microsekunden)

Aus diesen Pulsen werden drei verschiedene Kombinationen gebildet, die die folgende Bedeutung haben:

LLMM = Byte, diese Kombination geht jedem Byte vorraus
MMKK = Bit gesetzt = 1
KKMM = Bit nicht gesetzt = 0

Der Wert 65 wird also wie folgt auf Band aufgezeichnet:

LLMM	MMKK	KKMM	KKMM	KKMM	KKMM	KKMM	MMKK	KKMM	MMKK
BYTE	1	0	0	0	0	0	1	0	1
BIT	0	1	2	3	4	5	6	7	PARITY ODD

Dies ergibt eine Zeitspanne von 8.96 ms für ein Zeichen.

Beim Lesevorgang zählt ein Zähler von einen bestimmten Wert solange abwärts, wie ein Signal an der Leseleitung des Computers liegt. An dem erreichten Werten erkennt dann der Rechner, ob es sich um ein gesetztes Bit, ein nicht gesetztes Bit oder Bytemarke handelt.

Damit der Zähler aber immer zur richtigen Zeit gestartet

wird, ist eine genaue Synchronisierung zwischen Band und Zähler notwendig. Aus diesem Grunde geht jedem Block eine Synchronisation und ein Countdown voraus.

Ein Block ist folgendermaßen aufgebaut:

1. Synchronisationsbytes und Count Down
   Durch diesen Teil werden die Lesebausteine auf das Band synchronisiert. Weiterhin enthält die Synchronisation auch die Informationen, um was für einen Block es sich handelt.

2. Daten

3. Prüfsummenbyte
   Während des Abspeicherns und Ladens bildet der Rechner aus allen Bytes eine EXOR-Prüfsumme, indem er den letzten Wert mit dem nächsten durch eine logische "Exclusive-Oder" Funktion verknüpft und abspeichert. Wenn Lesefehler auftreten, stimmt diese Prüfsumme nicht mit der abgespeicherten überein.

4. Wiederholung der Daten
   Diese Wiederholung wird beim Lesevorgang mit den schon eingelesenen Bytes verglichen, um eventuelle Lesefehler festzustellen.

5. Prüfsummenbyte (wie 3)

6. Blockendmarkierung

Die kompletten Files werden wie folgt abgespeichert:

Programmfiles	Datenfiles
1. Header	Header
2. Programm	Datenblock evtl. weitere Datenblöcke
3. evtl. EOT-Block	evtl. EOT-Block

9. APPEND VON BASICPROGRAMMEN

Sie haben sich bestimmt schon darüber geärgert, daß das Commodore Basic 2.0 keine Befehle zur Verfügung stellt, um mehrere Programme miteinander im Speicher zu verknüpfen.

Wieder einmal kommen uns die Basic-Zeiger 43, 44 / 45, 46 zu Hilfe. Sie "verbiegen" den Basic-RAM-Startzeiger auf das Ende des im Speicher befindlichen Programms und laden das anzuhängende Programm nach. Anschließend muß der Zeiger 43, 44 wieder auf den alten Wert gebracht werden.

Im einzelnen müssen Sie wie folgt vorgehen:

1. Laden Sie das erste Programm ein. Die Zeilennummern müssen alle kleiner sein, als die des anzuhängenden Programms. Sind sie es nicht, arbeitet das zusammengefügte Programm nicht so, wie es eigentlich sollte.

2. Stellen Sie durch

   ```
   PRINT PEEK(43),PEEK(44)
   ```

   die Basic-Startadresse fest und notieren Sie sie am besten.

3. Geben Sie nun

   ```
   POKE43,(PEEK(45)+256*PEEK(46)-2)AND255
   POKE44,(PEEK(45)+256*PEEK(46)-2)/256
   ```

   ein.

   Da am Ende eines Basicprogramms immer zwei Null-Bytes zur Endmarkierung stehen, müssen Sie den Endvektor um zwei Bytes vermindern (Siehe auch Kapitel 2.1). Nun haben Sie scheinbar kein Programm mehr im Speicher.

Dies können Sie überprüfen, wenn Sie LIST geben.

4. Laden Sie nun das zweite Programm ein.
   Wenn Sie jetzt LIST geben, erscheint nur das zweite Programm.

5. Schreiben Sie nun die alten Werte wieder in die Speicherstellen 43, 44.

Durch diese Prozedur haben Sie nun zwei Programme aneinander gefügt.

Eleganter geht es natürlich mit einem Maschinenprogramm. Das folgende kleine Programm erkennt durch ein (@), dem Klammeraffen, als erstes Zeichen im Filenamen, daß das zu ladene Programm an das im Speicher befindliche angehängt werden soll.

PROGRAMMBESCHREIBUNG

Durch die Initialisierung wird der Ladevektor auf dieses Programm verschoben. Dadurch wird bei jeder Ladeoperation zu diesem Programm verzweigt.

In diesem Programm wird dann das erste Zeichen des Filenamens auf (@) getestet. Findet das Programm dieses Zeichen nicht, springt es in die alte Routine. Wird dieses Zeichen gefunden, wird in den Ladestartadressenvektor \$C3/\$C4 der Programmendvektor minus 2 übertragen und das Zeichen (@) im Filenamen gelöscht. Danach springt das Programm wieder in die alte Laderoutine.

```
25:     C000                       *=    $C000
30:     C000                       .OPT P1,O
32:     C000
32:     C000
33:                     ;******** MERGE **********
34:     C000
34:     C000                                  ;WERT IN KLAMMER GILT FUER
35:                     ;          VC 20
36:     C000
36:     C000
40:     0330            VECTOR   =    $330    ;LADEVEKTOR
50:     F4A5            LADEN    =    $F4A5   ;($F549) LADEROUTINE
60:     00BB            FILEADR  =    $BB     ;ZEIGER ZUM FILENAMEN
70:     00B7            FILELNG  =    $B7     ;FILENAMENLAENGE
80:     00C3            STARTADR =    $C3     ;ZEIGER AUF PRG. START
90:     002D            PROGEND  =    $2D     ;ZEIGER AUF PRG. ENDE
92:     C000
92:     C000
93:     C000
93:     C000
95:                     ;INITIALISIERUNG DES
96:                     ;    PROGRAMMS
97:     C000
97:     C000
100:    C000 A9 0B      INIT     LDA  #<START
110:    C002 A2 C0               LDX  #>START
120:    C004 8D 30 03            STA  VECTOR
130:    C007 8E 31 03            STX  VECTOR+1
140:    C00A 60                  RTS
142:    C00B
142:    C00B
144:                    ; HAUPTPROGRAMM
146:    C00B
146:    C00B
150:    C00B 48         START    PHA
160:    C00C A2 00               LDX  #0
170:    C00E A1 BB               LDA  (FILEADR,X)
180:    C010 C9 40               CMP  #"@"    ;ERSTES ZEICHEN '@'
190:    C012 D0 12               BNE  ENDE    ;NEIN => NORMAL LADEN
195:    C014 38                  SEC
200:    C015 A5 2D               LDA  PROGEND
210:    C017 E9 02               SBC  #2      ;ZEIGER UM 2 ERNIEDRIGEN
220:    C019 85 C3               STA  STARTADR  ;IN LADESTARTADR.
230:    C01B A5 2E               LDA  PROGEND+1
240:    C01D E9 00               SBC  #0
250:    C01F 85 C4               STA  STARTADR+1
260:    C021 18                  CLC
270:    C022 E6 BB               INC  FILEADR  ; '@' LOESCHEN
280:    C024 C6 B7               DEC  FILELNG
290:    C026 68         ENDE     PLA
300:    C027 4C A5 F4            JMP  LADEN
]C000-C02A
```

```
*************************************************************
   10 REM MERGE C64
*************************************************************
   20 E=256*PEEK(56)+PEEK(55)-1:A=E-41
   30 FOR I=A TO E:READ X:POKE I,X:NEXT
   40 READ X,Y,Z:X=A+X:Y=A+Y:Z=A+Z:H=INT(Z/256)
   45 POKE X,Z-256*H:POKE Y,H
   46 POKE E-1,PEEK(816):POKE E,PEEK(817)
   50 H=INT(A/256):POKE 56,H:POKE 55,A-256*H
   65 SYS A:NEW
32000 DATA 169,224,162,127,141,48,3,142,49,3,96,72,162,0,1
      61,187,201,64,208,18
32001 DATA 56,165,45,233,2,133,195,165,46,233,0,133,196,24,
      230,187,198,183,104
32002 DATA 76,165,244,1,3,11
```

```
*************************************************************
   10 REM  MERGE VC20
*************************************************************
   20 E=256*PEEK(56)+PEEK(55)-1:A=E-41
   30 FOR I=A TO E:READ X:POKE I,X:NEXT
   40 READ X,Y,Z:X=A+X:Y=A+Y:Z=A+Z:H=INT(Z/256)
   45 POKE X,Z-256*H:POKE Y,H
   46 POKE E-1,PEEK(816):POKE E,PEEK(817)
   50 H=INT(A/256):POKE 56,H:POKE 55,A-256*H
   65 SYS A:NEW
  100 DATA 169,224,162,127,141,48,3,142,49,3,96,72,162,0,1
      61,187,201,64,208,18
  110 DATA 56,165,45,233,2,133,195,165,46,233,0,133,196,24,
      230,187,198,183,104
  120 DATA 76,73,245,1,3,11
```

10. STEUERUNG DER DATASSETTE PER PROGRAMM

Bis jetzt habe ich nur über den Lade- und Speichervorgang mit Ihrer Datassette geschrieben. Dabei haben wir gesehen, daß das Band automatisch gestartet und gestoppt wurde. Der Computer besitzt also die Möglichkeit, den Motor zu steuern. Unter dem Motto, "was der Computer kann, kann ich schon lange", werde ich Ihnen jetzt beschreiben, wie Sie diese Steuerungsmöglichkeit in Ihrem Programm nutzen können.

Mit folgenden Speicherstellen können Sie den Motor steuern und abfragen, ob eine Taste am Recorder gedrückt wurde:

VC 20 Adr.	Wert	C-64 Adr.	Wert	Funktion
37148	252	1	AND 223	Motor an
37148	0	1	OR 32	Motor aus
192	0	192	0	Motor an
192	1	192	1	Motor aus
37151	64	1	16	Taste gedrückt

Auf eine Taste am Recorder warten:

VC 20	C-64
WAIT 37151,64,64	WAIT 1,16,16

Als Anwendungsbeispiel, wie Sie die Steuerungsmöglichkeiten nutzen können, folgt ein Programm, das Ihnen das Suchen eines Programms, das unter vielen auf einer Cassette abgespeichert ist, abnimmt. Es erstellt einen Katalog aller auf Ihrer Cassette gespeicherten Programme und findet ein bestimmtes Programm selbständig.

Das Programm arbeitet nach folgendem Prinzip:

Nachdem Sie ein oder mehrere Programme hinter dieses Beispielprogramm auf Cassette abgespeichert haben, stellt es mit Ihrer Hilfe und der Variable TI die Zeit fest, die der Cassettenrecorder benötigt, um vom Ende des Katalogprogrammes bis zum zu katalogisierenden Programm vorzuspulen. Dieser Zeitwert wird mit dem Filenamen des Programms in Datazeilen abgespeichert.

Soll später ein bestimmtes Programm gesucht werden, wird der Cassettenmotor beim Vorspulen solange angelassen, bis der aktuelle Zeitwert mit dem abgespeicherten übereinstimmt. Nach Drücken der PLAY-Taste wird dann das gewünschte File geladen.

So bedienen Sie das Programm:

Dieses Programm wird am Anfang der Cassette abgespeichert. Mit einem Abstand von ca. 10 Sekunden können Sie dahinter wie gewohnt die anderen Programme abspeichern. Sollen die Programme in den Katalog aufgenommen werden, notieren Sie sich die Startzählernummern und Filenamen der einzelnen Programme. Anschließend spulen Sie die Cassette an den Start zurück und laden das Katalogprogramm. Nach Starten des Programms wählen Sie den Punkt Neueingabe.

Zuerst werden Sie gebeten, die entsprechenden Filenamen in der richtigen Reihenfolge einzugeben. Sie werden dann vom Programm aufgefordert, den schnellen Vorlauf am Recorder einzustellen. Nun können Sie den Recorder durch Drücken der SPACE - Taste starten und bei den notierten Zählerständen genauso stoppen, wieder starten und beim nächsten Filestart stoppen, bis Sie so allen eingegebenen Programmen einen Zeitwert zugeordnet haben. Als letztes speichern Sie das Katalogprogramm wieder am Cassettenanfang ab.

Nun können Sie komfortabel ein im Katalog gespeichertes Programm nach der Angabe "LADEN" aus einem Bildschirmmenue

wählen. Bei gedrückter FFWD spult der Rechner bis zum gewünschten Programm. Nach Drücken der PLAY- und SPACE- Taste wird das Programm geladen.

PROGRAMMBESCHREIBUNG

In den Zeilen 120 bis 170 sind die Unterprogramme zum Starten und Stoppen des Motors. Soll das Programm auf einem VC 20 laufen, müssen die REM-Anweisungen in den Zeilen 130 und 160 gestrichen werden. Weiterhin müssen Sie in den Zeilen 120 und 150 hinter POKE 192,0: ein REM einfügen.

In den Zeilen 180 bis 340 wird das Programm initialisiert und das Menue auf den Bildschirm geschrieben. Bei der Initialisierung werden die Namens- und Zeitfelder dimensioniert und die Repeat-Funktion für alle Tasten eingeschaltet. Mit Hilfe der Betriebssystem-Routine SCREEN (65517) wird der Computertyp festgestellt und dementsprechend der Start vom Video- und Farbram gesetzt. Zuletzt werden die Variablen zur Motorsteuerung entsprechend dem Computer belegt (Zeilen 240 und 260). Aufgrund der Eingabe wird zur "Neueingabe" oder zum "Laden" verzweigt.

In den Zeilen 350 bis 660 befindet sich der Block "Neueingabe". In der Zeile 390 wird die Anzahl der schon eingegebenen Files über READ aus den Datazeilen eingelesen. In den Zeilen 400 - 420 werden die neuen Filenamen eingelesen und im Feld N$(n) abgespeichert.

In den Zeilen 530 bis 580 wird die Spulzeit zwischen den Startpunkten der einzelnen Programmen festgestellt und, den eingegebenen Filenamen entsprechend zugeordnet, im Feld ZS(n) gespeichert. Vorher wird in der Zeile 580 ein Offset von 10 abgezogen, um die Auslaufzeit des Motors auszugleichen. Aus gleichem Grunde wird bei der Summation der einzelnen Differenzzeiten der Zeitwert 10 addiert.

Um Fehlbedienungen zu vermeiden wird in Zeile 420 getestet, ob eine Taste am Recorder gedrückt ist und der Anwender ggf. gebeten die STOP-Taste zu betätigen.

In den Zeilen 600 - 660 werden Filenamen und Zeiten in Datazeilen geschrieben und das Programm beendet.

In Zeile 670 beginnt der Programmpunkt "Laden". Die Datenanzahl und die Filedaten werden eingelesen und in den Zeilen 720 - 820 werden die Files, je 8 Stück pro Seite, auf dem Bildschirm mit einem (>), das über die Zeilen 770 - 810 gesteuert wird, ausgedruckt. Ist die Abfrage nach der Taste F7 in Zeile 790 positiv, wird nach Zeile 830 zur Such- und Laderoutine verzweigt.

Nach dem Test, ob eine Taste am Recorder gedrückt ist (840) wird in der Zeile 900 wird mit dem WAIT - Befehl auf das Drücken der F.FWD-Taste am Recorder gewartet und dann, entsprechend der abgespeicherten Zeit der Cassettenmotor angelassen (Zeile 920 - 940).

In den Zeilen 950 - 1010 wird das gewünschte Programm geladen. Würde das Programm vom Programmodus her geladen, könnten nur Programme geladen werden, die kleiner als das Katalogprogramm sind. Um das zu vermeiden, wird der LOAD - Befehl, bzw. der FastTape LOAD-Befehl L auf den Bildschirm und RETURN in den Tastaturpuffer geschrieben. Somit wird das Programm aus dem Direktmodus geladen und die Basic-Vektoren entsprechend dem zu ladenen Programm gesetzt.

```
10 GOTO 190
20 ********************************
30 *                              *
40 *         K A T A L O G        *
50 *                              *
60 *         FUER CASSETTE        *
70 *                              *
80 ********************************
90 :REM
*******************************************************
100 REM          WARTESCHLEIFE
*******************************************************
110 POKE 198,0:WAIT 198,1:GET A$:RETURN
120 POKE 192,0:POKE 1,PEEK(1)AND 223:REM MOTOR AN C-64
130 :REM POKE 37148,252 :REM MOTOR AN VC20
140 RETURN
150 POKE 192,1:POKE 1,PEEK(1)OR 32:REM MOTOR AUS C-64
160 :REM POKE 37148,0 :REM MOTOR AUS VC20
170 RETURN
*******************************************************
180 REM "          INITIALISIERUNG
*******************************************************
190 HD$=" * * K A T A L O G * *"
200 SYS 65517:SP=PEEK(781):ZE=PEEK(782)
    :REM BILDSCHIRMDATEN HOLEN
210 BS=1024:BF=55296:FW=1:IF SP>25 THEN 260
220 BS=4*(PEEK(36866)AND 128)+64*(PEEK(36869)AND 120)
230 BF=37888+4*(PEEK(36866)AND 128)
240 WZ=37151:WW=64:REM VC20 WERTE 'WARTEN AUF RECORDE
    RTASTE
250 FW=6:GOTO 270
260 WZ=1:WW=16:REM C-64 WERTE 'WARTEN AUF RECORDERTAS
    TE
270 DIM N$(50),ZS(50):REM NAME UND STARTPOSITION
280 P1=3*SP+1:P2=2*SP
290 POKE 650,128:REM TASTENWIEDERHOLUNG
300 PRINT"[CLEAR]";:PRINT TAB(SP/2-11)HD$:PRINT"[DOWN7]"
310 PRINT TAB(SP/2-6)"[RVS]N[RVOFF]EUEINGABE":PRINT
```

```
320 PRINT TAB(SP/2-6)"[RVS]L[RVOFF]ADEN"
330 GOSUB 110:IF A$="L"THEN 680
340 IF A$<>"N"THEN 330
**********************************************************
350 REM ******** NEUEINGABE ************
**********************************************************
360 PRINT"[CLEAR,DOWN2]DIE EINGABE DER FILES ":PRINT
370 PRINT"MUSS SUKKSESSIV ERFOL";:IF SP<40 THEN PRINT
    :PRINT
380 PRINT"GEN.    @@ = ENDE"
390 RESTORE:READ A:ZA=A
400 INPUT"[DOWN]FILENAMEN";N$
410 IF N$<>"@@"THEN N$(A)=N$:A=A+1:GOTO 400
420 ZE=A:A=ZA:IF(PEEK(WZ)AND WW)=WW THEN 450
430 PRINT"[CLEAR]DRUECKEN SIE DIE"
    :PRINT"[DOWN,RVS]STOP[RVOFF;SPACE]TASTE"
440 PRINT"[DOWN]AM RECORDER":WAIT WZ,WW,255-WW
450 PRINT"[CLEAR]DRUEKEN SIE DIE ";:IF SP<40 THEN PRINT
    :PRINT
460 PRINT"[RVS]F.FWD[RVOFF,SPACE]TASTE!"
470 PRINT:PRINT"STARTEN SIE DEN RECOR-";
    :IF SP<40 THEN PRINT:PRINT:PRINT" ";
480 PRINT"[LEFT]DER DURCH DRUECKEN"
490 PRINT:PRINT"EINER TASTE UND ";:IF SP<40 THEN PRINT
    :PRINT
500 PRINT"STOPPEN SIE IHN AM "
510 PRINT:PRINT"START DES GEWUENSCH-";
    :IF SP<40 THEN PRINT:PRINT:PRINT" ";
520 PRINT"[LEFT]TEN FILES AUF GLEICHE":PRINT"WEISE."
530 WAIT WZ,WW,WW:GOSUB 150:DI=0
540 GOSUB 110:GOSUB 120:R=TI:REM CC AN, ZEIT MERKEN
550 PRINT"[CLEAR,DOWN5]STOPPEN BEI START:"
560 PRINT"[DOWN]"N$(A)
570 GOSUB 110:GOSUB 150:DI=TI-R+DI
    :REM CC AUS, ZEITDIFFERENZ BERECHNEN
580 ZS(A)=DI-10:DI=DI+10:A=A+1
590 IF A<ZE THEN 540:REM LETZTER FILENAME ?
**********************************************************
600 REM * ABSPEICHERN IN DATA ZEILEN **
**********************************************************
```

```
  610 PRINT"[CLEAR,DOWN3]2000DATA"A
      :REM ANZAHL IN DATAZEILE
  620 FOR I=ZA TO ZE-1:PRINT 2100+I"DATA"N$(I)","ZS(I)
  630   NEXT:PRINT"GOTO650"
  640 POKE 198,11:FOR I=0 TO 10:POKE 631+I,13:NEXT
      :PRINT"[HOME]":END
  650 PRINT"[CLEAR]SPULEN SIE JETZT ZU-"
      :PRINT"[DOWN]RUECK UND SPEICHERN"
      :PRINT"[DOWN]DAS PROGRAMM AM ANFANG"
  660 PRINT:PRINT"AB":GOSUB 120:END
************************************************************
  670 REM ******* LADEN DER FILES *******
************************************************************
  680 RESTORE:READ A:FOR I=0 TO A-1:READ N$(I),ZS(I):NEXT
  690   PRINT"[CLEAR]";:PRINT TAB(SP/2-11)HD$
  700   PRINT"[DOWN3,SPACE2]F1 = AUFW."
        :PRINT"[DOWN2,SPACE2]F2 = ABW."
x = 3   :PRINT"[DOWN2,SPACE2]F7 = LADEN"
  710   PRINT"[DOWN4]":PRINT TAB(SP/2-6)"[RVS,SPACE2]S P A
         C E  [RVOFF]":GOSUB 110:ZA=0
  720   PRINT"[CLEAR]";:PRINT TAB(SP/2-11)HD$:Z=0:PRINT
  730   FOR I=ZA TO ZA+8:IF N$(I)<>""THEN PRINT
          :PRINT TAB(3)N$(I)
  740     NEXT
  750   POKE BS+P1+Z*P2,62:POKE BF+P1+Z*P2,FW
  760   GOSUB 110:REM TASTATUR ABFRAGE
  770   IF(ASC(A$)=133)AND(Z>0)THEN POKE BS+P1+Z*P2,32
        :Z=Z-1:GOTO 750
  780   IF(ASC(A$)=134)AND(Z<8)AND(N$(ZA+Z+1)<>"")THEN POK
        E BS+P1+Z*P2,32:Z=Z+1:GOTO 750
  790   IF ASC(A$)=136 THEN 830
  800   IF ASC(A$)=133 AND Z=0 AND ZA THEN ZA=ZA-9:GOTO 720
  810   IF(ASC(A$)=134)AND(N$(I))<>""AND(Z=8)THEN ZA=ZA+9
        :GOTO 720
  820   GOTO 750
************************************************************
  830   REM               LADEN
************************************************************
  840   IF(PEEK(WZ)AND WW)=WW THEN 870
  850   PRINT"[CLEAR]DRUECKEN SIE DIE"
```

```
      :PRINT"[DOWN,RVS]STOP[RVOFF,SPACE]TASTE"
 860  PRINT"[DOWN]AM RECORDER":WAIT WZ,WW,255-WW
 870  PRINT"[CLEAR]";:PRINT TAB(SP/2-11)HD$:PRINT:PRINT
 880  PRINT"DRUECKEN SIE DIE ":PRINT"[DOWN,SPACE4,RVS,SP
      ACE]F.FWD [RVOFF]"
 890  PRINT:PRINT"TASTE"
 900  WAIT WZ,WW,WW:REM WARTEN AUF TASTE AM RECORDER
 910  PRINT" OK "
 920  R=TI:R2=R+ZS(Z+ZA)
 930  IF R2<TI THEN 950
 940  GOTO 930
 950  GOSUB 150
 960  PRINT"[CLEAR]";:PRINT TAB(SP/2-11)HD$
 970  PRINT:PRINT"DRUECKEN SIE DIE "
      :PRINT"[DOWN,SPACE6]PLAY    "
 980  PRINT:PRINT"TASTE"
 990  PRINT"[DOWN4]":PRINT TAB(SP/2-6)"[RVS,SPACE2]S P A
       C E  [RVOFF]":GOSUB 110
1000  PRINT"[CLEAR,DOWN3]←L";CHR$(34)(N$(Z+ZA))CHR$(34)
1010  POKE 198,1:POKE 631,13:POKE 37148,252:PRINT"[HOME]"
      :END
*********************************************************
1020  REM ****** PROGRAMMANZAHL ********
*********************************************************
2000  DATA O
*********************************************************
2005  REM ****** PROGRAMMDATAS *********
```

11. HARDWARE DER DATASSETTE

Bis jetzt habe ich nur die Software zur Datassettenhandhabung beschrieben. In diesem Kapitel möchte ich auf die Hardware der Datassette selbst eingehen.

11.1 PFLEGE DER DATASSETTE

Wie jeder andere Cassettenrecorder braucht die Datassette auch eine gewisse Pflege. Das häufige Auftreten eines "?LOAD ERROR" ist meistens die Folge eines verschmutzten, dejustierten oder magnetisierten Kopfes.

1. Reinigung der Köpfe und der Andruckrolle

   Hierzu eignen sich sehr gut handelsübliche Wattestäbchen. Sind sie nicht greifbar, können Sie auch ein Streichholz oder ein anderes Holzstäbchen an einer Seite mit Watte umwickeln. Als Reinigungsmittel empfiehlt sich Alkohol, Fleckenbenzin oder Spiritus benutzen.

   Zur Reinigung selbst öffnen Sie den Cassettenschacht und drücken die PLAY-Taste. Nun können Sie mit den mit Reinigungsmittel getränkten Wattestäbchen die Köpfe (Lese- /Schreibkopf und Löschkopf) reinigen, indem Sie solange über die Frontseite der Köpfe fahren, bis die Watte keine braune Färbung mehr annimmt, und anschließend mit einem trockenen Wattestäbchen nachreiben.

   Es empfiehlt sich, im gleichen "Abwasch" auch die Gummi-Andruckrolle zu reinigen. Nach längerem Betrieb der Datassette bildet sich nämlich ein schlüpfriger Belag aus Bandabrieb. Dadurch wird das Band nicht mehr gleichmäßig am Tonkopf vorbeigeführt wird. Zur Reinigung legen Sie ein Stück nicht flusenden Stoffs

(Taschentuch oder ähnliches) über den Zeigefinger, befeuchten es mit einem oben beschriebenen Reinigungsmittel und halten den Finger bei auf PLAY laufender Datassette von rechts gegen die Andruckrolle.

2. Demagnetisierung

Zur Demagnetisierung benutzen Sie einen käuflichen Demagnetisator oder eine Demagnetisierungscassette, wie sie für Audioanlagen erhältlich sind. Die Handhabung ist dann genauso, wie sie in der entsprechenden Bedienungsanleitung beschrieben wird.

11.2 WAHL UND HANDHABUNG DER CASSETTEN

An eine Cassette zur Daten- und Programmspeicherung braucht man nicht die Anforderungen zu stellen wie an eine Audiocassette. Sie brauchen also für Ihre Datassette keine Cassetten mit besonders gutem Frequenzgang oder großem Rauschabstand. Nach dem bisher gesagtem wäre die billigste Cassette gerade gut genug.

Der einzig wichtige Punkt ist, daß die Cassette keine "DROP OUTS" hat. DROP OUTS sind Stellen auf dem Band, an denen die Beschichtung mit magnetisierbaren Partikeln nicht einwandfrei ist. An diesen Stellen kann das Band nicht ordnungsgemäß magnetisiert werden, sodaß dort gespeicherte Informationen verlorengehen. Diese Gefahr ist leider bei normalen Superbilligcassetten gegeben. Meiner Meinung nach fahren Sie deshalb am besten mit den einfachen Cassetten irgend eines renommierten Herstellers.

Um zu lange Spulzeiten zu vermeiden, empfehle ich Ihnen maximal C60-Cassetten zu verwenden.

Wie Sie wissen, sind am Anfang oder am Ende des Bandes Vorspannbänder angebracht, um die Belastung, die beim

Anschlag an das Ende auf das Band einwirkt, abzufangen. Trotzalledem wird das Magnetband am Anfang und am Ende immer noch stärker belastet als an anderen Stellen, was zu einer Dehnung führen kann. Ich empfehle Ihnen deshalb, auf den ersten und letzten 10 - 20 Sekunden zur besseren Datensicherung keine Daten zu speichern.

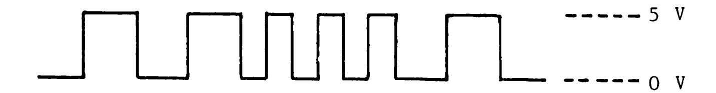
Sonst gilt für Datencassetten das gleiche wie für Audio- und Videocassetten. Setzen Sie sie keinen starken Magnetfeldern aus und lagern Sie sie staubgeschützt. Auch sollten Sie sie nicht zu lange lagern (über viele Monate), ohne sie einmal umzuspulen. Sonst treten die magnetischen Schichten des Bandes möglicherweise miteinader in Wechselwirkung und können sich dadurch verändern.

Um Programme gegen äußere Einflüsse oder auch eigenes Fehlverhalten zu schützen, empfiehlt es sich, von allen wichtigen Daten und Programmen eine Kopie anzufertigen, mit der Sie nicht arbeiten und die Sie auf einer schreibgeschützten Cassette ablegen. So können Sie immer dann, wenn Ihre Arbeitskopie zerstört wird, auf das Mutterfile zurückgreifen.

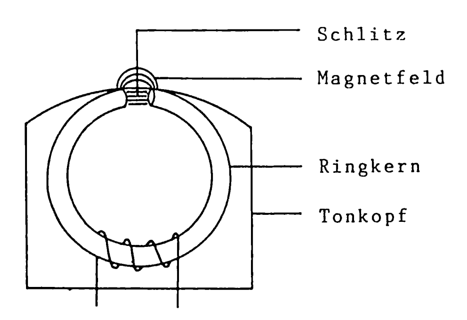
11.3 ARBEITSWEISE DER DATASSETTE

Wie im Kapitel 8 beschrieben wird die zu speichernde Information in Form von Pulsen der Datassette übermittelt. Dies geschieht dadurch, daß beim C-64 am Portbit 3 des Prozessors (Speicherstelle eins) entweder eine Spannung von ca. 5 V = 1 oder 0 V = 0 anliegt. Bei VC 20 ist es das Bit 3 der Speicherstelle 37152. Sowohl beim VC 20 als auch beim C-64 ist dieser Anschluß direkt mit den Cassettenport - Anschlüssen E und 5 verbunden.

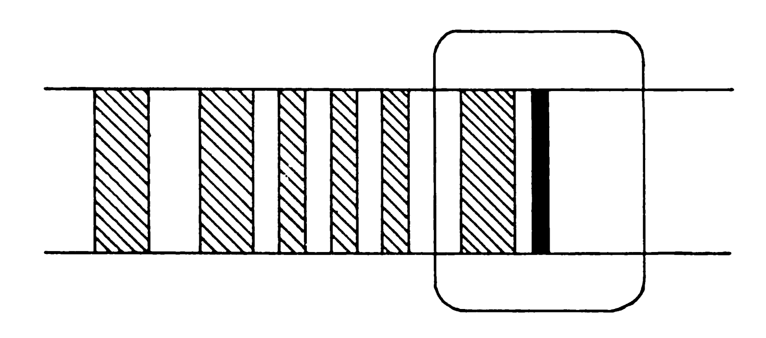
Ideal liegt das Signal folgendermaßen dort an:



Über einige elektronische Stufen, die ich hier nicht näher erläutern will, gelangt das Signal zum Tonkopf. Der Tonkopf ist eine Spule mit Ringkern, die einen kleinen Schlitz auf der dem Band zugewandten Seite hat.



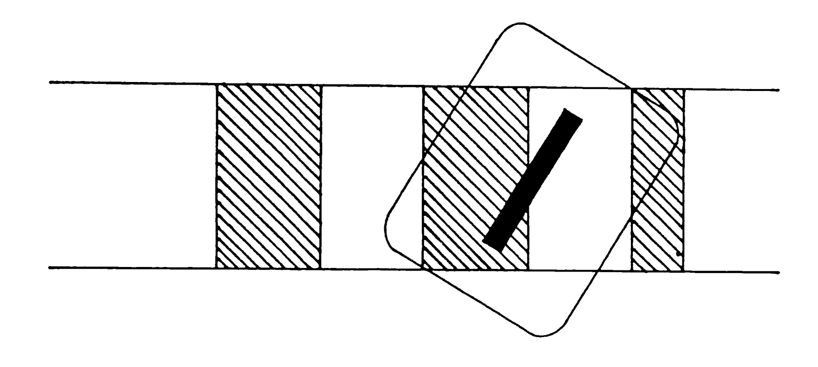
Entsprechend der an den Spulenanschlüssen angelegten Spannung entsteht ein Magnetfeld am Schlitz, das dann die magnetisierbaren Partikel auf dem Band in eine bestimmte Richtung ausrichtet. Das idealisiertes Magnetiesierungs-muster auf dem Band sieht dan folgendermaßen aus:



Wenn Sie nun das Band lesen, wird der ganze Vorgang umgekehrt.

Durch die magnetisierten Partikel wird über den Tonkopfschlitz eine Spannung in der Spule erzeugt, die über einige Baustufen wieder zu dem gleichen Spannungsverlauf führt wie beim Abspeichern. Diese Spannung liegt an den Cassettenport-Anschlüssen D und 4 an.

Wenn nun aus irgendwelchen Gründen die Stellung des Tonkopfes zum Band beim Lesen anders ist als beim Speichern, erhalten wir folgendes Bild:



Im oberen Teil des Schlitzes ist die Magnetisierungsrichtung eine andere als im unteren Teil. Beide Wirkungen löschen

sich gegenseitig, und die Daten werden fehlerhaft eingelesen.

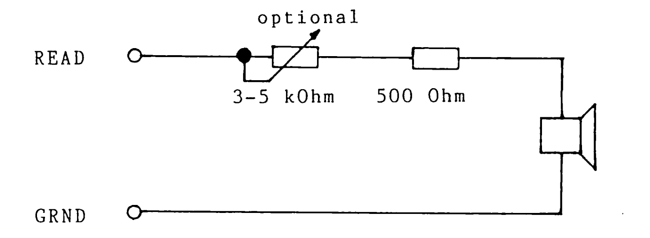
11.4 EIN LAUTSPRECHER FÜR DIE DATASSETTE

Sie haben sich bestimmt schon darüber geärgert, daß Sie außer dem ziemlich unzuverlässigen Zählwerk keine Möglichkeit haben, die Position Ihres Bandes festzustellen. Wünschenswert wäre, festzustellen, ob Sie gerade innerhalb eines Programms das nächste suchen, oder ob Sie gerade am Start eines Programms sind. So könnten Sie viel schneller ein bestimmtes Programm suchen.

Diese Möglichkeit können Sie sich durch den Anschluß eines Lautsprechers an die Datassette schaffen. Mit Hilfe dieses Lautsprechers hören Sie, wo ein Programm anfängt. Sie können nun durch die Wiederholung von kurzzeitigem, schnellem Vorlauf und folgendem "Reinhören" mit PLAY viel besser den Start des folgenden Programms finden.

Weiterhin ist das akustische "feed-back" bei der Tonkopfjustierung, das ich in Kapitel 11.5. beschreiben werde, ein wichtiges Hilfsmittel.

Zum Anschluß an die Datassette eignet sich jeder Lautsprecher. Da es ja nicht auf einen möglichst guten Klanggenuß ankommt, können Sie auch aus einem alten Miniradio den Lautsprecher ausbauen. Dieser Lautsprecher wird an die READ-Leitung angeschlossen. Damit Sie diesen Anschluß nicht überlasten, müssen Sie noch einen Widerstand von ca. 500 Ohm in Reihe schalten. Weiterhin können Sie noch ein Potentiometer von 2 - 5 KOhm in Reihe zum Lautsprecher schalten, um die Lautstärke regeln zu können.



Anschluß des Lautsprechers

Am einfachsten läßt sich ein Lautsprecher an den Stecker der Datassette anschließen. Dazu öffnen Sie den Stecker, der an den Computer gesteckt wird, und klemmen den einen Anschluß an die GROUND-Leitung (A/1) und den anderen an die READ-Leitung (D/4).

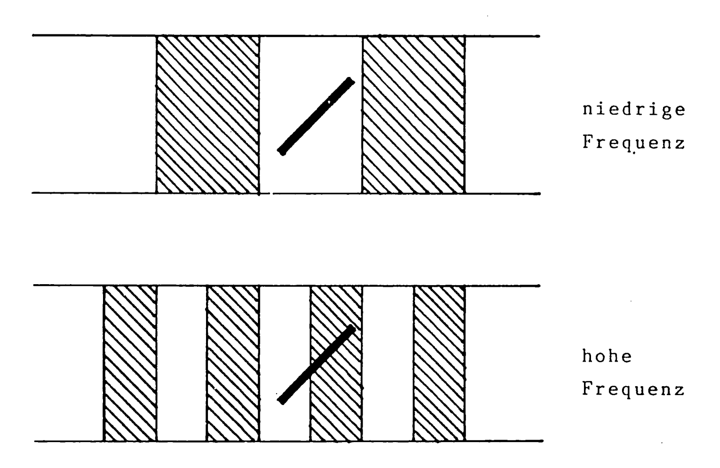
Schließen Sie dann den Stecker wieder und stecken ihn an den Cassetten-Port Ihres Computers.

Schalten Sie nun den Computer ein, legen die bespielte Cassette ein und drücken die PLAY-Taste. Jetzt hören Sie entweder einen eintönigen Pfeifton oder ein zirpendes Geräusch. Der Pfeifton entspricht der Synchronisation und kennzeichnet einen Blockanfang. Das zirpende Geräusch entspricht den aufgezeichneten Daten.

11.5 KOPFJUSTAGE

Es kann aus verschiedenen Gründen vorkommen, daß der Tonkopf nicht mehr in der richtigen Stellung zum Band steht, was zum fehlerhaften Laden führt. Dies kann z.B. passieren, wenn Sie Programme von einer Cassette einlesen wollen, die nicht mit Ihrer Datassette bespielt wurde. Es kommt aber auch vor, daß sich der Kopf selbständig nach längerer Zeit dejustiert.

Mit Hilfe eines Lautsprechers ist die Dejustierung gut zu hören. Wenn der Kopf gegenüber dem Band auch nur leicht verwinkelt ist, geht dies als erstes zu Lasten der hohen Frequenzen. Ich möchte das anhand zweier Zeichnungen einmal verdeutlichen:



Wie aus den Abbildungen zu sehen ist, gibt es bei niedrigen Frequenzen immer noch Bandpositionen, bei denen über die ganze Spaltänge die gleiche Magnetisierung wirkt.

Bei hohen Frequenzen liegt am Spalt immer nur eine Mischung verschieder magnetisierter Bereiche an. Somit werden solche Frequenzen gedämpft.

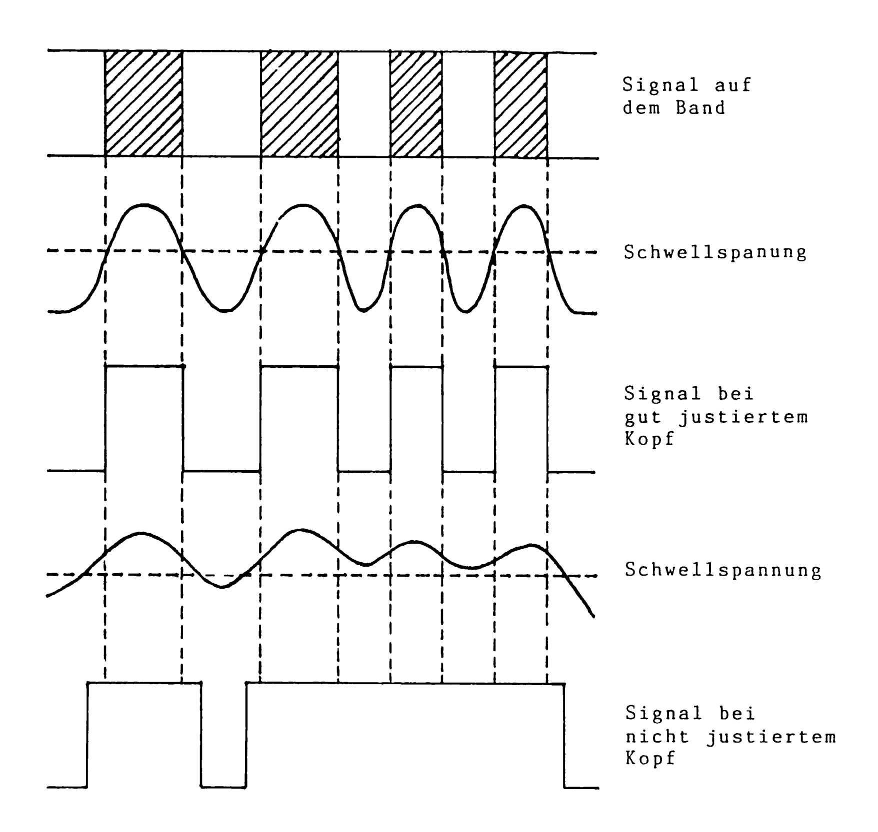


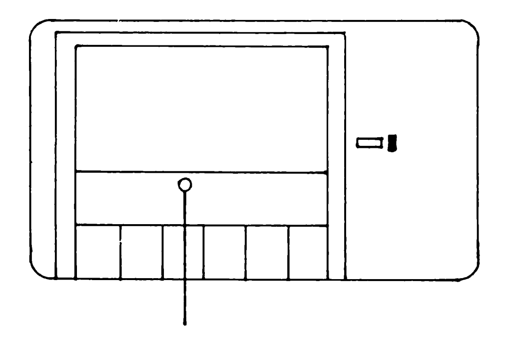
Bild 2 : Ausgangs-Signale bei verschieden Tonkopfstellungen

Beim Datenlesen erhalten Sie das in Abbildung 2 dargestellte Bild. Das vom Tonkopf aufgenommene Signal ist annähernd sinusförmig. Dieses Signal wird innerhalb der Datassette über eine Schmitt-Trigger Schaltung in computerlesbare Rechteckimpulse umgesetzt. Ein Schmitt-Trigger ist nichts anderes als ein Schalter, der sich bei einer bestimmten Spannungen, der sogenannten "Schwellenspannung", ein- bzw. ausschaltet und somit eine Rechteckspannung am Ausgang erzeugt.

Aus der Abbildung 2 wird klar, daß durch einen dejustierten Kopf die Zeitverhältnisse zwischen den einzelnen Pulsen, die ja die Information enthalten, nicht mehr stimmen. Deshalb wird das Band fehlerhaft gelesen.

Nach dieser theoretischen Einführung kommen wir zum praktischen Teil.

Wenn Sie die neuere Ausführung der Datassette haben, finden Sie ungefähr in der Mitte vor dem Cassettenfach ein kleines Loch.

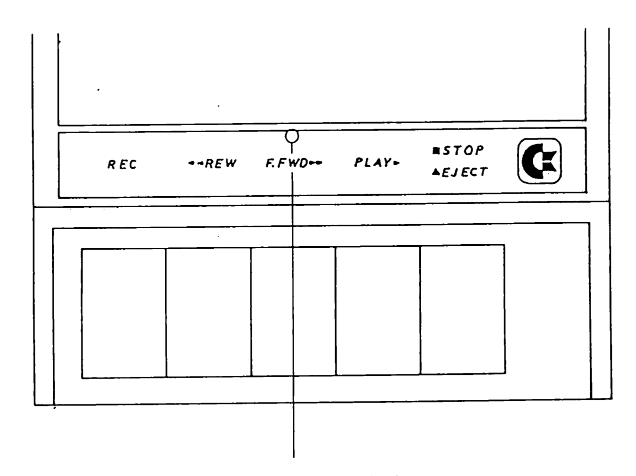


Loch zur Tonkopfjustierschraube

Durch dieses Loch erreichen Sie mit einem kleinen Kreuzschlitzschraubendreher bei gedrückter PLAY-Taste die

Tonkopfeinstellschraube.

Bei den alten Datassetten gibt es kein Loch. Um bei diesen Geräten den Tonkopf justieren zu können, müssen Sie selber ein Loch in das Plastikgehäuse bohren. Die genaue Position dieses Lochs entnehmen Sie bitte der folgenden Zeichnung.



Hier müssen Sie das Loch bohren

Doch bevor wir nun beginnen, den Kopf zu justieren, noch zwei Warnungen:

1. Es ist möglich, daß Sie den Kopf so sehr dejustieren, daß eine richtige Justage ohne aufwendige Hilfsmittel (Meßgeräte) nur sehr schwer möglich ist. Auch müssen Sie darauf achten, daß sie die Justierschraube nicht gänzlich herausdrehen. Normalerweise reichen maximal zwei Umdrehungen in jede Richtung aus, um den Kopf zu justieren. Meist reicht schon eine halbe Umdrehung aus.

2. Nachdem Sie den Kopf neu justiert haben, kann es sehr gut sein, daß sie die Programme, die Sie mit der letzten Einstellung abgespeichert haben, nicht mehr laden können.

Damit Sie möglichst leicht die ursprüngliche Kopfstellung wiederfinden können und einen dejustierten Kopf besser justieren können, benutzen Sie das folgende kleine Programm. Es schreibt ein Zirpen auf Cassette mit einem großen Anteil hoher Frequenzen.

Da es ja die hohen Frequenzen sind, die zuerst gedämpft werden, erleichtern Sie sich durch dieses Zirpen die Einstellung auf die Kopfstellung, mit der das Zirpen abgespeichert wurde, sehr. Auch können Sie damit den Kopf bedeutend genauer justieren.

Geben Sie also das Programm ein und speichern es ab. Schreiben Sie dann auf eine leere Cassette ca. zwei Minuten das von diesem Programm erzeugte Zirpen.

```
************************************************************
   10 REM KOPFJUSTAGE HILFSPROGRAMM C64
************************************************************
   20 E=256*PEEK(56)+PEEK(55)-1:A=E-17
   30 FOR I=A TO E:READ X:POKE I,X:NEXT
   40 PRINT"STARTADRESSE"A
   50 PRINT:PRINT"LEGEN SIE EINE LEERCASSETTE EIN UND"
   60 PRINT:PRINT"DRUECKEN SIE RECORD & PLAY"
   70 WAIT 1,16,16
   80 PRINT:PRINT"ENDE DURCH RUN/STOP TASTE"
   90 SYS A:END
 1000 DATA 160,255,169,255,162,255,32,177,251,136,208,246,
      32,225,255,208,239,96
```

```
************************************************************
   10 REM KOPFJUSTAGE HILFSPROGRAMM VC 20
************************************************************
   20 E=256*PEEK(56)+PEEK(55)-1:A=E-17
   30 FOR I=A TO E:READ X:POKE I,X:NEXT
   40 PRINT"STARTADRESSE"A
   50 PRINT:PRINT"LEGEN SIE EINE LEERCASSETTE EIN UND"
   60 PRINT:PRINT"DRUECKEN SIE RECORD & PLAY"
   70 WAIT 37151,64,64
```

```
80 PRINT:PRINT"ENDE DURCH RUN/STOP TASTE"
90 SYS A:END
1000 DATA 160,255,169,255,162,255,32,177,251,136,208,246,
     32,225,255,208,239,96
```

Spulen Sie das Band zurück, nehmen Sie den kleinen Kreuzschlitzschraubendreher zur Hand und drücken die PLAY-Taste. Führen Sie nun durch das beschriebene Loch den Schraubendreher senkrecht ein, bis er in der Schraube einrastet. Drehen sie nun jeweils eine halbe Umdrehung nach rechts und links von der Ausgangsposition und achten Sie dabei auf die typische Klangveränderung. Nur in einer Stellung, nämlich in der Ausgangsstellung, hören Sie die meisten hohen Frequenzen.

Legen Sie nun eine bespielte Datencassette ein und wiederholen Sie den Vorgang. Sie werden jetzt in einem größeren Bereich keine Klangveränderung mehr feststellen. Die beste Stellung zwischen den hörbaren Änderungen ist hier die Mittelstellung.

Nachdem Sie sich auf diese Weise etwas "eingehört" haben, können Sie auf gleiche Weise den Kopf auch auf Programme justieren, die Sie nicht laden konnten. Versuchen Sie auch hier nach Gehör die beste Kopfstellung herauszufinden. Danach versuchen Sie, das Programm zu laden. Läßt es sich immer noch nicht laden, verändern Sie die Kopfposition etwas und versuchen es erneut.

Gelingt es Ihnen trotz vielfachen Versuchens nicht, das Programm einwandfrei zu laden, verfahren Sie so, wie es in Kapitel 4.1 beschrieben wurde.

Ehe Sie aber das UNNEW-Programm laden, bzw. das Programmfragment abspeichern, müssen Sie den Tonkopf mit Hilfe der von Ihnen angefertigten Justiercassette wieder in die Ausgangsposition bringen.

11.6 ANDERE CASSETTENRECORDER ZUR DATENSPEICHERUNG

In meinen Ausführungen habe ich mich immer auf die Datassette von Commodore bezogen. Es ist aber ohne weiteres möglich, andere Cassettenrecorder mit einer entsprechenden Anpassung als Aufzeichnungsgerät zu verwenden. Die Anpassung erfolgt durch ein Interface, welches die vom Band kommenden Signale in für den Rechner verständliche Rechteckimpulse umsetzt. Solche Interface-Bausteine werden von verschiedenen Herstellern recht preiswert angeboten.

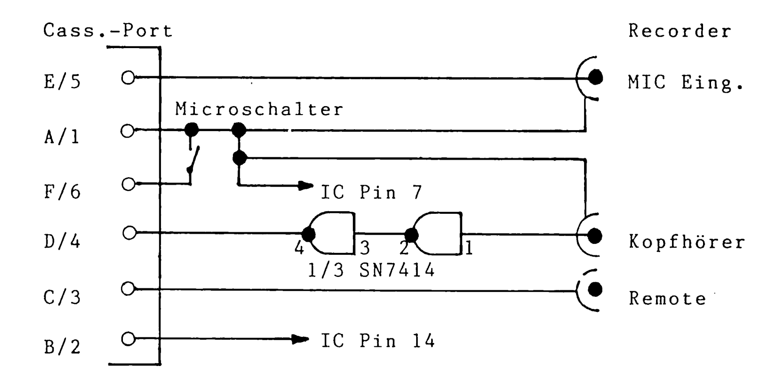
Der Vollständigkeit halber möchte ich hier eine kleine Schaltung zum Selbstbau darstellen. Die benötigten Bauteile entnehmen Sie bitte folgender Aufstellung.

Bauteile für ein Cassettenrecorder-Interface

1 Platinen - Stecker für den Cassettenport am VC 20/C64
1 Stecker für den Kopfhörerausgang Ihres Cassettenrecorders
1 Stecker für den Microphoneingang Ihres Cassettenrecorders
1 IC SN 7414 (6 * Invertierender Schmitt-Trigger)
1 Microschalter (ein/aus)
ggf.
1 Stecker für den Remote-Eingang Ihres Recorders.

Als Recorder eignen sich für diese Schaltung nur solche, die mit einer Betriebsspannung von 4 - 6 V arbeiten und den Minuspol auf Masse haben.

Die Verschaltung der Bauteile finden Sie auf der nächsten Seite.



Beschreibung der Anschlüsse am Cassetten Port:

A/1 = Masse
B/2 = +5V
C/3 = Cassetten-Motor (+5 V = ein)
D/4 = Leseleitung
E/5 = Schalterleitung für Recordertaste

In dem IC SN 7414 sind sechs invertierende Schmitt-Trigger enthalten. Schmitt-Trigger sind Schaltungen, die aus einem beliebigen Spannungsverlauf entsprechende Rechteckimpulse bilden wie ich es im Kapitel 11.5 beschrieben habe.

Von diesen sechs benötigen wir für diese Schaltung zwei, da es sich ja um invertierende Schaltungen handelt.

Die Betriebsspannung dieses Bausteins entnehmen wir den Cassetten-Port (Pin7 - A/1, Pin14 - B/2).

Wenn Sie den Cassettenrecorder nur für Ihren Rechner benutzen, können Sie die Schaltung in das Batteriefach einbauen, und die Microphon- und Kopfhörerleitung festanlöten. Damit sparen Sie die Klinkenstecker.

Sehr praktisch wäre es dann noch, wenn Sie in der Recordermechanik eine Stelle finden würden, wo Sie den

Microschalter so anbringen können, daß er bei dem Drücken einer Taste geschlossen wird. Geht das nicht, müssen Sie den Schalter außen anbringen und manuell betätigen, bevor Sie eine Recorder-Taste drücken.

TEIL 2

12. EIN NEUES CASSETTENBETRIEBSSYSTEM - FASTTAPE

Im ersten Teil habe ich Ihnen die Handhabung der Datassette mit dem im Rechner eingebauten Betriebssystem erklärt. Dazu habe ich Ihnen einige Programme gezeigt, die das Arbeiten mit Ihrem Datenrecorder vereinfachen. Nun möchte ich Ihnen ein neues Betriebssystem vorstellen, das nicht nur die schon beschriebenen Vereinfachungen beinhaltet, sondern auch noch ca. 10 - 20 mal schneller arbeitet, und damit eine Diskettenstation noch überflügelt.

Ich muß zugeben, daß Sie sich aufgrund der Programmlänge die Finger fast wund tippen können, aber das Ergebnis lohnt die Arbeit. Das Programm hat folgende Charakteristika:

1. 10 mal schnelleres Speichern und Laden der Programme.

2. Unterstützung der Datenspeicherung.
   Daten können bis zu 20 mal schneller geladen und gespeichert werden als mit dem normalen Betriebssystem.

3. Append von Basic-Programmen ist mit <u>einem</u> Befehl möglich.

4. Das Laden in oder das Speichern von bestimmten Speicherbereichen ist ebenfalls mit <u>einem</u> Befehl möglich.

5. Programme und Daten, die mit FastTape abgespeichert wurden, können sowohl vom VC 20 als auch vom C-64 gelesen werden.

6. Alle FastTape Befehle arbeiten sowohl im Direktmodus als auch innerhalb von Programmen.

DIE BEDIENUNG DES PROGRAMMS

Nach dem Laden des Programms initialisieren sie es mit SYS7*4096 beim VC 20 und mit SYS12*4096 beim C-64. Durch den Basic-Lader wird das Programm automatisch initialisiert. Nach einem RESET müssen Sie das Programm aber neu initialisieren. Jedem FastTape Befehl geht ein Pfeil nach links (←) voraus.

Ihnen stehen jetzt folgende Befehle zur Verfügung (In Klammern stehende Ausdrücke sind optional, müssen also nicht gegeben werden):

SA = Sekundäradresse
Sadr, Eadr = Start- bzw. Endadresse
"name" kann auch als Stringvariable übergeben werden

S("NAME",SA,SADR,EADR) *FASTTAPE SAVE*

S("name") : Basic-Programm wird als verschiebbares File im Programm gespeichert und mit L an den Basic-RAM-Start geladen.

S"name",1 : Basic-Programm wird als absolutes File gespeichert und mit L an die Adresse geladen, von wo es abgespeichert wurde.

S"Name",1,Sadr,Eadr : Der Speicherbereich zwischen Sadr und Eadr wird als absolutes File gespeichert.

L("NAME",SA,SADR) *FASTTAPE LOAD*

Wird dieser Befehl innerhalb eines Programms gegeben, startet danach das Basic-Programm analog zum LOAD-Befehl wieder bei der ersten Zeile.

L("name") : Ein Programm oder Speicherbereich, der mit S abgespeichert wurde, wird entsprechend der bei S gegebenen SA entweder absolut oder verschiebbar geladen. Wird dieser Befehl im Direktmodus gegeben, werden die Basic-Vektoren entsprechend dem geladenen Programm gesetzt.

L"name",1 : Programme und Speicherbereiche werden absolut geladen. Auch im Direktmodus werden durch diesen Befehl die Basic-Vektoren nicht geändert. Dadurch ist kein NEW-Befehl nach dem Laden eines Maschinenprogramms notwendig.

L"name,1,Sadr : Ein Speicherbereich wird an Sadr geladen, unabhängig davon, von welcher Adresse es abgespeichert wurde. Dadurch haben Sie z.B. die Möglichkeit, einen einmal abgespeicherten Bildschirminhalt auch dann wieder in den Videorom einzulesen, wenn Sie den Videorom verschoben haben.

V("NAME".SA.SADR) *FASTTAPE VERIFY*

Arbeitet analog zu L, vergleicht aber nur den Speicherinhalt mit den auf Band stehenden Bytes.

M("NAME") *FASTTAPE MERGE*

Mit diesem Befehl können Sie ein FastTape-Programm an das im Speicher befindliche anhängen.

DS "NAME" *DATASAVE*

Dieser Befehl speichert alle bis dahin definierten Variablen, Felder und Strings ab. Mit diesem Befehl werden nur die im Stringspeicher stehenden Strings abgespeichert. Wenn in Ihrem Programm Stringzuweisungen wie

```
100 AS="TEST"
```

existieren, können Sie die entsprechenden Strings nur mit dem gleichen Programm wieder richtig laden. Abhilfe können Sie durch eine Verknüpfung mit einem Leerstring schaffen. Hier gilt das gleiche, wie im Kapitel 5.1 beschrieben.

DL "NAME" *DATALOAD*

Mit diesem Befehl laden Sie die mit DS abgespeicherten Variablen, Felder und Strings. Alle bis zu diesem Befehl definierten Variablen, Felder und Strings werden gelöscht.

Achtung!!

DS und DL muß immer in Verbindung mit einem Filenamen verwandt werden. Durch diese Befehle werden die Variablen, Felder und Strings in drei Blöcken abgespeichert bzw. geladen. Der Filename dient dem Computer zur Identifizierung des jeweils ersten Blocks.

Achtung!!

Das Basic-RAM-Ende muß bei DS und dem entsprechenden DL das gleiche sein. Die Stringtabelle wird als absoluter Block gespeichert und geladen. Wenn Sie das Ende des Basic-RAMs heruntersetzen, nachdem Sie DS gegeben haben und dann mit DL Daten laden, kann eventuell ein hinter das neue Basic-RAM-Ende geschriebene Maschinenprogramm zerstört werden.

Damit Sie sehen, wie komfortabel Sie mit diesen Befehlen Datenverarbeitung betreiben können, finden Sie im nächsten Kapitel, ein ausführlich beschriebenes Dateiverarbeitungsprogramm. Durch den Basic-Lader FT64 wird das Programm im C-64 ab der Speicherstelle $C000 abgelegt.

Der Basic-Lader FT20 legt das Programm im VC 20 ab der Speicherstelle $7000 ab, d.h. dieses Programm arbeitet nur dann in einem VC 20, wenn er voll ausgebaut ist. Es ist aber ohne weiteres möglich, mit Hilfe eines MONITOR-Programms und dem ASSEMBLER-Listing das FastTape-Programm auf einen anderen Speicherbereich umzuschreiben.

```
82:     C000                    .PAG 61
90:     C000                    .OPT P2
92:     C000
92:     C000
94:                     ; FASTTAPE FUER VC20 UND C-64
95:     C000
95:     C000
95:     C000
96:                     ;WERTE IN KLAMMERN GELTEN FUER
97:     C000
97:     C000
98:                     ;           VC 20
100:    C000                    *=    $C000
110:    0000            COMPU    =    $00      ; (2)
120:    0001            PORT     =    $1       ; ($9120)
130:    0002            ABSFLG   =    $2       ;
140:    00C0            MOFLAG   =    $C0
150:    00A3            BITC     =    $A3
153:    0004            BEFANZ   =    4
155:    005F            CODE     =    "_"
157:    0073            CHRGET   =    $73
160:    0308            VECTOR   =    $308     ; BEFEHLSADR. HOLEN
165:    02A0            IRQFLG   =    $2A0     ; IRQ FLAG
170:    0090            STATUS   =    $90      ; STATUSBYTE
175:    00C3            STARTV   =    $C3      ; LADESTARTVEKTOR
180:    00AE            ENDVEC   =    $AE      ; LADEENDVEKTOR
185:    00D7            PSUMME   =    $D7      ; PRUEFSUMMENBYTE
186:    033C            CASPUF   =    $033C    ; STARTADR. VOM CASSETTENPUFFER
187:    C000
187:    DD00            CIA1     =    $DD00    ; C-64 = STARTADRESSE DES CIA1
188:    DC00            CIA2     =    $DC00    ; C-64 = STARTADRESSE DES CIA2
189:    C000
189:    C000
190:    F68F            SAVING   =    $F68F    ;($F728) GIBT SAVING 'NAME' AUS
195:    F5D2            LOADING  =    $F5D2    ;($F66A) GIBT LOADING AUS
200:    A82C            STOP     =    $A82C    ; ($C82C) TESTET STOPTASTE
220:    FC93            ENDEN    =    $FC93    ; ($FCCF) LADEN BEENDEN
230:    E17A            BEFEND   =    $E17A    ; ($E177) RUECKSPRUNG VOM LADEN
240:    F750            FOUND    =    $F750    ; ($F7D3) GIBT FOUND 'NAME' AUS
260:    F5A9            STOEND   =    $F5A9    ; ($F641) ENDADR. NACH X/Y
300:    A7E7            OLD      =    $A7E7    ; ($C7E7) BEFEHL AUSWERTEN
310:    A7AE            INTER    =    $A7AE    ; ($A7AE) INTERPRETERSCHLEIFE
320:    F838            RPTASTE  =    $F838    ; ($F8B7) WARTET AUF PLAY-TASTE
330:    F817            PTASTE   =    $F817    ; ($F894) WARTET AUF  TASTE
335:    B526            GABCOL   =    $B526    ; ($D526) GARBAGE COLLECTION
340:    E206            WZEICH   =    $E206    ; ($E203) WEITERE ZEICHEN PRINT
350:    E257            HFNAM    =    $E257    ; ($E254) FILENAMEN HOLEN
360:    E200            HOLPAR   =    $E200    ; ($E1FD) FILEPARAMETER HOLEN
365:    E20E            CHRIN    =    $E20E    ; ($E20B) ZEICHEN HOLEN
370:    AF08            SYNTAX   =    $AF08    ; ($CF08) SYNTAX ERROR AUSGEBEN
380:    AD8A            FRMNUM   =    $AD8A    ; ($CD8A) TERM AUSWERTEN
390:    B7F7            FACINT   =    $B7F7    ; ($D7F7) IN INTEGER WANDELN
395:    A660            CLEAR    =    $A660    ; ($C660) CLR
400:    A437            FEHLAUS  =    $A437    ; ($C437) FEHLER AUSGEBEN
410:    A677            BEENDEN  =    $A677    ; ($C677)
420:    C000
420:    C000
430:    FFB7            HOLSTA   =    $FFB7    ; STATUS HOLEN
440:    FFBD            SETNAM   =    $FFBD    ; FILENAMEPARAMETER SETZEN
```

```
450:    FFBA            SETLFS    =    $FFBA    ; FILEPARAMETER SETZEN
460:    FFE4            GET       =    $FFE4    ; EIN ZEICHEN EINLESEN
500:    C000
500:    C000
510:    C000
510:    C000
520:                    ;PROGAMM INITIALISIEREN
530:    C000
530:    C000
560:    C000 A9 0B      INIT      LDA  #<START
570:    C002 8D 08 03             STA  VECTOR
580:    C005 A9 C0                LDA  #>START
590:    C007 8D 09 03             STA  VECTOR+1
600:    C00A 60                   RTS
602:    C00B
602:    C00B
603:    C00B
603:    C00B
604:                    ; PROGRAMMSTART
606:    C00B
606:    C00B
610:    C00B 20 73 00 START       JSR  CHRGET   ; ZEICHEN HOLEN
620:    C00E F0 04                BEQ  ENDE
630:    C010 C9 5F                CMP  #CODE    ; MIT FASTTAPECODE
640:    C012 F0 03                BEQ  FTAPE    ; VERGLEICHEN
650:    C014 4C E7 A7 ENDE        JMP  OLD      ; UNGLEICH => ALTE ROUTINE
654:    C017
654:    C017
655:                    ;FASTTAPE BEFEHL AUSWERTEN
657:    C017
657:    C017
660:    C017 20 73 00 FTAPE       JSR  CHRGET
670:    C01A 20 20 C0             JSR  SUCH
680:    C01D 4C AE A7             JMP  INTER
690:    C020 A2 00      SUCH      LDX  #0
700:    C022 DD 54 C0 SUCHL       CMP  TAB,X    ; MIT BEFEHLSBUCHSTABEN
710:    C025 F0 08                BEQ  BEFEHL   ; AUS TABELLE VERGLEICHEN
720:    C027 E8                   INX
730:    C028 E4 04                CPX  BEFANZ
740:    C02A D0 F6                BNE  SUCHL
750:    C02C 4C 08 AF             JMP  SYNTAX   ; KEIN FASTTAPE BEFEHL
755:    C02F
755:    C02F
760:    C02F 8A         BEFEHL    TXA
770:    C030 0A                   ASL
770:    C031 AA                   TAX           ; BEFZAHL * 2
780:    C032 BD 5A C0             LDA  BEFADR+1,X  ; BEFEHLSADRESSE AUF
790:    C035 48                   PHA           ; STACK
800:    C036 BD 59 C0             LDA  BEFADR,X
800:    C039 48                   PHA
810:    C03A 4C 73 00             JMP  CHRGET   ; UND BEFEHL AUSFUEHREN
815:    C03D
815:    C03D
820:    C03D C9 53      DATA      CMP  #"S"     ; DATEN SPEICHERN
830:    C03F F0 07                BEQ  DS
840:    C041 C9 4C                CMP  #"L"     ; DATEN LADEN
850:    C043 F0 09                BEQ  DL
860:    C045 4C 08 AF             JMP  SYNTAX
865:    C048
```

```
865:    C048
870:    C048 20 73 00 DS        JSR   CHRGET
870:    C04B 4C 11 C3           JMP   DASAV1
880:    C04E 20 73 00 DL        JSR   CHRGET
880:    C051 4C 61 C3           JMP   DATLOD
885:    C054
885:    C054
886:                    ;TABELLE DER BEFEHLSBUCHSTABEN
887:    C054
887:    C054
890:    C054 53 4C 56 TAB       .ASC "SLVMD"
895:    C059
895:    C059
896:                    ;TABELLE DER BEFEHLSADRESSEN
897:    C059
897:    C059
900:    C059 62 C0    BEFADR    .WORDSAVE-1
900:    C05B 6F C1              .WORDLOAD-1
900:    C05D 72 C1              .WORDVERIFY-1
910:    C05F 61 C1              .WORDMERGE1-1
910:    C061 3C C0              .WORDDATA-1
920:    C063
920:    C063
1650:   C063
1650:   C063
1660:   C063
1660:   C063
1665:                   ;SAVE ROUTINE
1666:   C063
1666:   C063
1670:   C063 A2 05    SAVE      LDX   #$05      ; ANZAL DER SYNCHR.
1680:   C065 86 AB              STX   $AB       ; WIEDERHOLUNGEN
1682:   C067 A2 00              LDX   #$00      ; FLAG LOESCHEN
1684:   C069 86 02              STX   ABSFLG
1690:   C06B 20 B0 C2           JSR   GETPARA    ; PARAMETER HOLEN
1692:   C06E A5 02              LDA   ABSFLG
1694:   C070 29 02              AND   #2        ; TEST AUF BIT 2
1695:   C072 D0 0B              BNE   ABSOLUT    ; GESETZT => ABSOLUT LADEN
1700:   C074 A2 04              LDX   #$04
1710:   C076 B5 2A    LOOP1     LDA   $2A,X     ; SPEICHERT BASICSTART UM
1720:   C078 95 AB              STA   $AB,X
1725:   C07A 95 A6              STA   $A6,X
1730:   C07C CA                 DEX
1740:   C07D D0 F7              BNE   LOOP1
1750:   C07F 20 38 F8 ABSOLUT   JSR   RPTASTE
1760:   C082 20 8F F6           JSR   SAVING    ; 'SAVING NAME' AUSGEBEN
1770:   C085 20 FC C0           JSR   MOTOR     ; MOTOR EINSCHALTEN
1774:   C088
1774:   C088
1775:                   ;VORSPANN SCHREIBEN
1776:   C088
1776:   C088
1780:   C088 20 13 C1           JSR   SYNCH     ; SYBCHRONISATION 1 SCHREIBEN
1790:   C08B A5 B9              LDA   $B9       ; SECUNDAERADR. INCREMENTIEREN
1800:   C08D 18                 CLC
1810:   C08E 69 01              ADC   #$01
1820:   C090 CA                 DEX
1830:   C091 20 33 C1           JSR   WBYTE     ; SCHREIBT SA
1840:   C094 A2 08              LDX   #$08+COMPU
```

```
1850:   C096 B9 A7 00 LOOP2     LDA   $A7,Y    ; SCHREIBT START- UND ENDADRESSE
1860:   C099 20 33 C1           JSR   WBYTE
1870:   C09C A2 06              LDX   #$06+COMPU
1880:   C09E C8                 INY
1890:   C09F C0 05              CPY   #$05
1900:   C0A1 EA                 NOP
1910:   C0A2 D0 F2              BNE   LOOP2
1920:   C0A4 A0 00              LDY   #$00     ;SCHREIBT FILENAMEN UND 'SPACES'
1930:   C0A6 A2 04              LDX   #$04+COMPU
1940:   C0A8 B1 BB     LOOP3    LDA   ($BB),Y
1950:   C0AA C4 B7              CPY   $B7
1960:   C0AC 90 03              BCC   TEXT
1970:   C0AE A9 20              LDA   #$20
1980:   C0B0 CA                 DEX
1990:   C0B1 20 33 C1 TEXT      JSR   WBYTE
2000:   C0B4 A2 05              LDX   #$05
2010:   C0B6 C8                 INY
2020:   C0B7 C0 BB              CPY   #$BB
2030:   C0B9 D0 ED              BNE   LOOP3
2034:   C0BB
2034:   C0BB
2035:                  ;SYNCHRONISATION 2 SCHREIBEN
2036:   C0BB
2036:   C0BB
2040:   C0BB A9 02              LDA   #$02
2050:   C0BD 85 AB              STA   $AB
2060:   C0BF 20 13 C1           JSR   SYNCH    ; SCHREIBT SYNCHRONISATION
2070:   C0C2 98                 TYA
2080:   C0C3 20 33 C1           JSR   WBYTE    ; 0 BYTE => ENDE VORSPANN
2090:   C0C6 84 D7              STY   PSUMME   ; PRUEFSUMMENBYTE LOESCHEN
2100:   C0C8 A2 07              LDX   #$07+COMPU
2104:   C0CA
2104:   C0CA
2105:                  ;PROGRAMM SCHREIBEN
2106:   C0CA
2106:   C0CA
2110:   C0CA EA                 NOP
2120:   C0CB B1 AC     PRGLOOP  LDA   ($AC),Y  ; PROGRAMM AUF BAND SCHREIBEN
2130:   C0CD 20 33 C1           JSR   WBYTE
2140:   C0D0 A2 03              LDX   #$03+COMPU
2150:   C0D2 E6 AC              INC   $AC      ; PROGRAMMZEIGER ERHOEHEN
2160:   C0D4 D0 04              BNE   NOHI
2170:   C0D6 E6 AD              INC   $AD
2180:   C0D8 CA                 DEX
2190:   C0D9 CA                 DEX
2200:   C0DA A5 AC     NOHI     LDA   $AC
2210:   C0DC C5 AE              CMP   ENDVEC   ; PROGRAMMENDE EREICHT
2220:   C0DE A5 AD              LDA   $AD
2230:   C0E0 E5 AF              SBC   ENDVEC+1
2240:   C0E2 90 E7              BCC   PRGLOOP  ; NEIN => WEITER
2250:   C0E4 EA                 NOP
2260:   C0E5 A5 D7     LOOP4    LDA   PSUMME   ; PRUEFSUMME SCHREIBEN
2270:   C0E7 20 33 C1           JSR   WBYTE
2280:   C0EA A2 07              LDX   #$07+COMPU
2290:   C0EC 88                 DEY
2300:   C0ED D0 F6              BNE   LOOP4
2310:   C0EF C8                 INY
2320:   C0F0 84 C0              STY   MOFLAG   ; MOTOR AUSSCHALTEN VORBEREITEN
2330:   C0F2 58                 CLI
```

```
2340:   C0F3 18                 CLC
2350:   C0F4 A9 00              LDA   #$00
2360:   C0F6 8D A0 02           STA   IRQFLG
2370:   C0F9 4C 93 FC           JMP   ENDEN
2372:   C0FC
2372:   C0FC
2373:                   ;MOTOR STARTEN
2374:   C0FC
2374:   C0FC
2375:   C0FC 20 2C A8 MOTOR     JSR   STOP
2380:   C0FF A0 00              LDY   #$00
2390:   C101 84 C0              STY   MOFLAG
2400:   C103 AD 11 D0           LDA   $D011     ; BILDSCHIRM AUS
2410:   C106 29 EF              AND   #$EF      ; ENTFAELLT BEI
2420:   C108 8D 11 D0           STA   $D011     ; VC 20
2430:   C10B CA       ANLOOP    DEX             ; WARTET BIS MOTOR HOCH
2440:   C10C D0 FD              BNE   ANLOOP    ; GELAUFEN IST
2450:   C10E 88                 DEY
2460:   C10F D0 FA              BNE   ANLOOP
2470:   C111 78                 SEI
2480:   C112 60                 RTS
2481:   C113
2481:   C113
2482:                   ; SYNCHRONISATION GENERIEREN
2484:   C113
2484:   C113
2490:   C113 A0 00    SYNCH     LDY   #$00
2500:   C115 A9 02    LOOP5     LDA   #$02      ; STARTBYTE 255-9 MAL SCHREIBEN
2510:   C117 20 33 C1           JSR   WBYTE
2520:   C11A A2 07              LDX   #$07+COMPU   ; BYTEWERT
2530:   C11C 88                 DEY
2540:   C11D C0 09              CPY   #$09
2550:   C11F D0 F4              BNE   LOOP5
2560:   C121 A2 05              LDX   #$05+COMPU
2570:   C123 C6 AB              DEC   $AB
2580:   C125 D0 EE              BNE   LOOP5
2581:   C127
2581:   C127
2582:                   ; COUNTDOWN SCHREIBEN
2584:   C127
2584:   C127
2590:   C127 98       COUNTS    TYA             ; COUNTDOWN
2600:   C128 20 33 C1           JSR   WBYTE
2610:   C12B A2 07              LDX   #$07+COMPU
2620:   C12D 88                 DEY
2630:   C12E D0 F7              BNE   COUNTS
2640:   C130 CA                 DEX
2650:   C131 CA                 DEX
2660:   C132 60                 RTS
2662:   C133
2662:   C133
2664:                   ;BYTE AUF BAND SCHREIBEN
2666:   C133
2666:   C133
2670:   C133 85 BD    WBYTE     STA   $BD
2680:   C135 45 D7              EOR   PSUMME
2690:   C137 85 D7              STA   PSUMME
2700:   C139 A9 08              LDA   #$08
2710:   C13B 85 A3              STA   BITC
```

```
2720:  C13D 06 BD     SHIFT    ASL   $BD
2730:  C13F A5 01              LDA   PORT
2740:  C141 29 F7              AND   #$F7
2750:  C143 20 55 C1           JSR   T1LOOP
2760:  C146 A2 11              LDX   #$11+COMPU
2770:  C148 EA                 NOP
2780:  C149 09 08              ORA   #$08
2790:  C14B 20 55 C1           JSR   T1LOOP
2800:  C14E A2 0E              LDX   #$0E+COMPU
2810:  C150 C6 A3              DEC   BITC
2820:  C152 D0 E9              BNE   SHIFT
2830:  C154 60                 RTS
2840:  C155 CA        T1LOOP   DEX
2850:  C156 D0 FD              BNE   T1LOOP
2860:  C158 90 05              BCC   BIT0     ; WENN CARRY GESETZT,
2870:  C15A A2 0B              LDX   #$0B     ; DANN ZEITSCHLEIFE
2880:  C15C CA        T2LOOP   DEX            ; VERLAENGERN
2890:  C15D D0 FD              BNE   T2LOOP
2900:  C15F 85 01     BIT0     STA   PORT
2902:  C161 60                 RTS
2903:  C162
2903:  C162
2904:  C162
2904:  C162
2905:                 ;MERGE
2906:  C162
2906:  C162
2909:                 ;MERGE
2910:  C162 A5 2D     MERGE1   LDA   $2D
2911:  C164 38                 SEC
2912:  C165 E9 02              SBC   #2
2912:  C167 A8                 TAY
2912:  C168 A5 2E              LDA   $2E
2912:  C16A E9 00              SBC   #0
2913:  C16C A2 00              LDX   #0
2914:  C16E F0 09              BEQ   MERGE2
2915:  C170
2915:  C170
2916:  C170
2916:  C170
2917:                 ;LADEN
2918:  C170
2918:  C170
2919:  C170
2919:  C170
2920:  C170 A2 00     LOAD     LDX   #$00
2925:  C172 2C                 .BYTE$2C
2930:  C173 A2 01     VERIFY   LDX   #$01
2940:  C175 A4 2B              LDY   $2B      ; BASICSTARTVECTOREN IN
2950:  C177 A5 2C              LDA   $2C      ; STARTSPEICHERSTELLEN
2960:  C179 86 0A     MERGE2   STX   $0A      ; LOAD FLAG (0=LOAD/1=VERIFY)
2970:  C17B 86 93              STX   $93      ; SETZEN
2975:  C17D A2 00              LDX   #0
2980:  C17F 86 02              STX   ABSFLG   ; ABSOLUTFAG LOESCHEN
2985:  C181 84 C3              STY   STARTV
2990:  C183 85 C4              STA   STARTV+1
3000:  C185 20 B0 C2           JSR   GETPARA
3010:  C188 20 8E C1           JSR   LOADR    ; LOAD UND VERIFY
3020:  C18B 4C 7A E1           JMP   BEFEND
```

```
3022:  C18E
3022:  C18E
3024:                  ;LADEROUTINE
3026:  C18E
3026:  C18E
3040:  C18E 20 0A C2 LOADR    JSR  SSYNCH
3050:  C191 A5 AB             LDA  $AB
3060:  C193 C9 02             CMP  #$02     ; IST PRG ABS. GESPEICHERT
3070:  C195 F0 08             BEQ  ABSOL    ; => ABSOLUT LADEN
3080:  C197 C9 01             CMP  #$01     ; BYTE NICHT 1 => WEITERSUCHEN
3090:  C199 D0 F3             BNE  LOADR
3100:  C19B A5 B9             LDA  $B9      ; SEKUNDAERADRESSE = 0
3110:  C19D F0 10             BEQ  RELLOD   ; => VERSCHIEBLICH LADEN
3112:  C19F A9 02    ABSOL    LDA  #2       ; ABSOLUTFLAG SETZEN
3113:  C1A1 05 02             ORA  ABSFLG
3114:  C1A3 85 02             STA  ABSFLG
3120:  C1A5 AD 3C 03          LDA  CASPUF   ; STARTADRESSE AUS PUFFER
3130:  C1A8 85 C3             STA  STARTV   ; IN STARTVECTOR
3140:  C1AA AD 3D 03          LDA  CASPUF+1
3150:  C1AD 85 C4             STA  STARTV+1
3155:  C1AF 20 02 C3 RELLOD   JSR  ADTEST
3156:  C1B2 A5 02             LDA  ABSFLG   ; TEST AUF 'WARTEN'
3157:  C1B4 29 04             AND  #4
3158:  C1B6 D0 08             BNE  NOWAIT
3160:  C1B8 20 50 F7          JSR  FOUND    ; FOUND 'NAME' AUSGEBEN
3170:  C1BB 20 E4 FF WAIT     JSR  GET      ; ENTFAELLT BEI
3180:  C1BE F0 FB             BEQ  WAIT     ; VC 20
3190:  C1C0 20 2C A8 NOWAIT   JSR  STOP
3200:  C1C3 A4 B7             LDY  $B7      ; FILENAMENLAENGE
3210:  C1C5 F0 0B             BEQ  NONAME
3220:  C1C7 88       TESTNA   DEY           ; FILENAMENTESTEN
3230:  C1C8 B1 BB             LDA  ($BB),Y
3240:  C1CA D9 41 03          CMP  CASPUF+5,Y
3250:  C1CD D0 BF             BNE  LOADR
3260:  C1CF 98                TYA
3270:  C1D0 D0 F5             BNE  TESTNA
3280:  C1D2 84 90    NONAME   STY  STATUS   ; STATUS LOESCHEN
3290:  C1D4 20 D2 F5          JSR  LOADING
3300:  C1D7 AD 3E 03          LDA  CASPUF+2  ; ENDADRESSE BERECHNEN
3310:  C1DA 38                SEC           ; AUS DIFFERENZ START- + ENDADR.
3320:  C1DB ED 3C 03          SBC  CASPUF   ; AUS DEM HEADER
3330:  C1DE 08                PHP           ; PLUS BESTIMMTER
3340:  C1DF 18                CLC           ; STARTADRESSE
3350:  C1E0 65 C3             ADC  STARTV
3360:  C1E2 85 AE             STA  ENDVEC
3370:  C1E4 AD 3F 03          LDA  CASPUF+3
3380:  C1E7 65 C4             ADC  STARTV+1
3390:  C1E9 28                PLP
3400:  C1EA ED 3D 03          SBC  CASPUF+1
3410:  C1ED 85 AF             STA  ENDVEC+1
3420:  C1EF 20 1F C2          JSR  PLOAD
3430:  C1F2 A5 BD             LDA  $BD
3440:  C1F4 45 D7             EOR  PSUMME   ; PRUEFSUMME TESTEN
3450:  C1F6 05 90             ORA  STATUS
3460:  C1F8 F0 04             BEQ  OK
3470:  C1FA A9 FF             LDA  #$FF
3475:  C1FC 85 90             STA  STATUS
3480:  C1FE A5 02    OK       LDA  ABSFLG   ; FLAG TESTEN
3484:  C200 D0 03             BNE  ALTADR   ; WENN GESETZT=> ALTE ENDVECTORE
```

```
3486:   C202 4C A9 F5          JMP  STOEND   ; ERHALTEN
3488:   C205 A6 2D     ALTADR  LDX  $2D
3490:   C207 A4 2E             LDY  $2E
3492:   C209 60                RTS
3500:   C20A 20 58 C2 SSYNCH   JSR  SCOUNT   ; SYNCHRONISATION SUCHEN
3510:   C20D C9 00             CMP  #$00     ; SYNCHR. VOR HEADER =>
3520:   C20F F0 F9             BEQ  SSYNCH   ; SYNCH NACH HEADER SUCHEN
3530:   C211 85 AB             STA  $AB      ; SEKUNDAERADRESSE + 1
3540:   C213 20 86 C2 SPSTART  JSR  HOLBYT   ; START DES PROGRAMMS
3550:   C216 91 B2             STA  ($B2),Y  ; SUCHEN
3560:   C218 C8                INY
3570:   C219 C0 C0             CPY  #$C0
3580:   C21B D0 F6             BNE  SPSTART
3590:   C21D F0 2D             BEQ  GEFUN    ; ENDE VORSPANN,PRG LADEN
3600:   C21F 20 58 C2 PLOAD    JSR  SCOUNT   ; PROGRAMM LADEN
3610:   C222 20 86 C2 LLOOP    JSR  HOLBYT
3620:   C225 C4 93             CPY  $93
3630:   C227 D0 02             BNE  VERGL    ; BEI VERIFY NUR VERGLEICHEN
3640:   C229 91 C3             STA  (STARTV),Y
3650:   C22B D1 C3     VERGL   CMP  (STARTV),Y
3660:   C22D F0 02             BEQ  GLEICH
3670:   C22F 86 90             STX  STATUS
3680:   C231 45 D7     GLEICH  EOR  PSUMME   ; PRUEFSUMME BERECHNEN
3690:   C233 85 D7             STA  PSUMME
3700:   C235 E6 C3             INC  STARTV   ; ADRESSE ERHOEHEN
3710:   C237 D0 02             BNE  NOTHI
3720:   C239 E6 C4             INC  STARTV+1
3730:   C23B A5 C3     NOTHI   LDA  STARTV
3740:   C23D C5 AE             CMP  ENDVEC   ; ENDADRESSE ERREICHT
3750:   C23F A5 C4             LDA  STARTV+1
3760:   C241 E5 AF             SBC  ENDVEC+1
3770:   C243 90 DD             BCC  LLOOP    ; NEIN => WEITER
3780:   C245 20 86 C2          JSR  HOLBYT
3790:   C248 20 FC C0          JSR  MOTOR
3800:   C24B C8                INY
3810:   C24C 84 C0     GEFUN   STY  MOFLAG
3820:   C24E 58                CLI
3830:   C24F 18                CLC
3840:   C250 A9 00             LDA  #$00
3850:   C252 8D A0 02          STA  IRQFLG
3860:   C255 4C 93 FC          JMP  ENDEN
3870:   C258 20 17 F8 SCOUNT   JSR  PTASTE   ; COUNTDOWN SUCHEN
3880:   C25B 20 FC C0          JSR  MOTOR
3890:   C25E 84 D7             STY  PSUMME
3900:   C260 A9 07             LDA  #$07     ; ($27) WERT FUER TIMER LO
3910:   C262 8D 06 DD          STA  CIA1+6   ; (PORT+8) IN TIMER LO
3920:   C265 A2 01             LDX  #$01     ; WERT FUER TIMER HI
3930:   C267 20 99 C2 SSTART   JSR  HOLBIT   ; START SUCHEN
3940:   C26A 26 BD             ROL  $BD
3950:   C26C A5 BD             LDA  $BD
3960:   C26E C9 02             CMP  #$02     ; START GEFUNDEN
3970:   C270 D0 F5             BNE  SSTART
3980:   C272 A0 09             LDY  #$09     ; => COUNTDOWN SUCHEN
3990:   C274 20 86 C2 ENDE2    JSR  HOLBYT   ; ENDE DER '2' - BYTES
4000:   C277 C9 02             CMP  #$02     ; SUCHEN
4010:   C279 F0 F9             BEQ  ENDE2
4020:   C27B C4 BD     COUNTL  CPY  $BD      ; TESTET DEN COUNTDOWN
4030:   C27D D0 E8             BNE  SSTART
4040:   C27F 20 86 C2          JSR  HOLBYT
```

```
4050:  C282 88              DEY
4060:  C283 D0 F6           BNE   COUNTL
4070:  C285 60              RTS
4072:  C286
4072:  C286
4075:                ; BYTE VON BAND HOLEN
4076:  C286
4076:  C286
4080:  C286 A9 08    HOLBYT  LDA   #$08     ; 8 BIT
4090:  C288 85 A3           STA   BITC
4100:  C28A 20 99 C2 SHIFT7  JSR   HOLBIT
4110:  C28D 26 BD           ROL   $BD      ; SCHIBT CARRY IN EINGABEPUFFER
4120:  C28F EA              NOP
4130:  C290 EA              NOP
4140:  C291 EA              NOP
4150:  C292 C6 A3           DEC   BITC
4160:  C294 D0 F4           BNE   SHIFT7
4170:  C296 A5 BD           LDA   $BD
4180:  C298 60              RTS
4182:  C299
4182:  C299
4184:                ; BIT VON BAND HOLEN FUER C-64
4185:  C299
4185:  C299
4190:  C299 A9 10    HOLBIT  LDA   #$10
4200:  C29B 2C 0D DC WAITDA  BIT   CIA2+13  ;WARTET AUF SIGNAL AN 'FLAG'
4210:  C29E F0 FB           BEQ   WAITDA
4220:  C2A0 AD 0D DD        LDA   CIA1+13  ; ICR LADEN. HAT TB 0 ERREICHT
4225:                ; => BIT 1 IST GEZT
4230:  C2A3 8E 07 DD        STX   CIA1+7   ; WERT IN TB HI
4240:  C2A6 48              PHA            ; ICR RETTEN
4250:  C2A7 A9 19           LDA   #$19     ; TB STARTEN, EINMALIGES ZAEHLEN
4260:  C2A9 8D 0F DD        STA   CIA1+15
4270:  C2AC 68              PLA            ; ICR HOLEN
4280:  C2AD 4A              LSR   A        ; BIT 1 INS CARRY SCHIEBEN
4290:  C2AE 4A              LSR   A
4300:  C2AF 60              RTS
4310:  C2B0
4310:  C2B0
4320:                ; 'HOLBIT' FUER VC20
4330:  C2B0
4330:  C2B0
4340:                ;HOLBIT LDA #$02
4350:                ;WAITDA BIT PORT+13 ; IFR BIT 1 TESTEN
4360:                ;       BEQ WAITDA
4370:                ;       LDA PORT+13
4380:                ;       STX PORT+9 ; TIMER HI SETZEN
4390:                ;       BIT PORT+1
4400:                ;       ASL
4410:                ;       ASL
4420:                ;       ASL
4430:                ;       RTS
4432:  C2B0
4432:  C2B0
4435:                ;PARAMETER HOLEN
4436:  C2B0
4436:  C2B0
4450:  C2B0 A9 00    GETPARA LDA   #0
4460:  C2B2 20 BD FF        JSR   SETNAM
```

```
4470:  C2B5 A2 01              LDX  #1        ; DEFAULT FUER GA
4480:  C2B7 A0 00              LDY  #00       ; DEFAULT FUER SA
4490:  C2B9 20 BA FF           JSR  SETLFS
4510:  C2BC 20 06 E2           JSR  WZEICH    ; WEITERE ZEICHEN
4520:  C2BF 20 57 E2           JSR  HFNAM
4540:  C2C2 20 06 E2           JSR  WZEICH
4550:  C2C5 20 00 E2           JSR  HOLPAR
4560:  C2C8 8A                 TXA
4570:  C2C9 A8                 TAY
4580:  C2CA 20 BA FF           JSR  SETLFS
4600:  C2CD 20 06 E2           JSR  WZEICH
4610:  C2D0 20 F4 C2           JSR  HOLWERT
4620:  C2D3 86 AC              STX  $AC
4622:  C2D5 86 A7              STX  $A7
4624:  C2D7 84 A8              STY  $A8
4630:  C2D9 84 AD              STY  $AD
4632:  C2DB A2 01              LDX  #1        ; FLAG FUER AN BESTIMMTE
4634:  C2DD 86 02              STX  ABSFLG    ; ADRESSE LADEN UND RETTEN
4650:  C2DF 20 06 E2           JSR  WZEICH
4660:  C2E2 20 F4 C2           JSR  HOLWERT
4670:  C2E5 86 AE              STX  $AE
4672:  C2E7 86 A9              STX  $A9
4674:  C2E9 84 AA              STY  $AA
4680:  C2EB 84 AF              STY  $AF
4685:  C2ED A5 02              LDA  ABSFLG
4690:  C2EF 09 02              ORA  #2
4695:  C2F1 85 02              STA  ABSFLG
4700:  C2F3 60                 RTS
4710:  C2F4 20 0E E2 HOLWERT   JSR  CHRIN     ; NAECHSTEN WERT HOLEN
4720:  C2F7 20 8A AD           JSR  FRMNUM    ; AUSDRUCK AUSWERTEN
4730:  C2FA 20 F7 B7           JSR  FACINT    ; IN 2BYTE WERT UMRECHNEN
4740:  C2FD A6 14              LDX  $14
4750:  C2FF A4 15              LDY  $15
4760:  C301 60                 RTS
4770:  C302 A5 02    ADTEST    LDA  ABSFLG    ; FLAG LADEN
4780:  C304 29 01              AND  #1
4790:  C306 F0 08              BEQ  NORM      ; UNGLEICH => NORMAL LADEN
4800:  C308 A5 AC              LDA  $AC
4810:  C30A 85 C3              STA  STARTV
4820:  C30C A5 AD              LDA  $AD
4830:  C30E 85 C4              STA  STARTV+1
4840:  C310 60       NORM      RTS
4940:  C311
4940:  C311
4945:  C311
4945:  C311
4950:                ; DATASAVE
4970:  C311
4970:  C311
5000:  C311 20 26 B5 DASAV1    JSR  GABCOL
5010:  C314 A2 05              LDX  #5
5020:  C316 86 AB              STX  $AB
5030:  C318 20 B0 C2           JSR  GETPARA
5040:  C31B A9 01              LDA  #1        ; SA=1
5050:  C31D 85 B9              STA  $B9
5060:  C31F A2 04              LDX  #4
5070:  C321 B5 2C    ULOOP     LDA  $2C,X     ; VARIABLEN START- UND
5080:  C323 95 AB              STA  $AB,X     ; ENDADRESSEN UMSPEICHERN
5085:  C325 95 A6              STA  $A6,X
```

```
5090:   C327 CA                 DEX
5100:   C328 D0 F7              BNE   ULOOP
5110:   C32A 20 7F C0           JSR   ABSOLUT   ; ABSPEICHERN
5112:   C32D A2 00              LDX   #0
5114:   C32F 86 B7              STX   $B7       ; KEIN FILENAME
5115:   C331 A2 05              LDX   #5
5117:   C333 86 AB              STX   $AB
5119:   C335 CA                 DEX
5130:   C336 B5 2E     U2LOOP   LDA   $2E,X     ; ARRAY START- UND
5140:   C338 95 AB              STA   $AB,X     ; ENDADRESSEN UMSPEICHERN
5145:   C33A 95 A6              STA   $A6,X     ; ENDADRESSEN UMSPEICHERN
5150:   C33C CA                 DEX
5160:   C33D D0 F7              BNE   U2LOOP
5170:   C33F 20 7F C0           JSR   ABSOLUT   ; ABSPEICHERN
5175:   C342 A2 05              LDX   #5
5177:   C344 86 AB              STX   $AB
5190:   C346 A5 33              LDA   $33       ; STRING START- UND
5200:   C348 A6 34              LDX   $34       ;
5210:   C34A 85 AC              STA   $AC
5215:   C34C 85 A7              STA   $A7
5220:   C34E 86 AD              STX   $AD
5225:   C350 86 A8              STX   $A8
5230:   C352 A5 37              LDA   $37       ; ENDADRESSEN UMSPEICHERN
5240:   C354 A6 38              LDX   $38       ;
5250:   C356 85 AE              STA   $AE
5255:   C358 85 A9              STA   $A9
5260:   C35A 86 AF              STX   $AF
5265:   C35C 86 AA              STX   $AA
5270:   C35E 4C 7F C0           JMP   ABSOLUT   ; ABSPEICHERN
5300:   C361
5300:   C361
5305:   C361
5305:   C361
5310:                  ; DATALOAD
5320:   C361
5320:   C361
5410:   C361 20 B0 C2 DATLOD    JSR   GETPARA
5420:   C364 A9 01              LDA   #1        ; AN BESTIMMTE ADRESSE LADEN
5430:   C366 85 02              STA   ABSFLG
5440:   C368 A5 2D              LDA   $2D       ; VARIABLENSTARTADRESSE ALS
5450:   C36A A6 2E              LDX   $2E       ; STARTADRESSE
5460:   C36C 85 AC              STA   $AC       ; UMSPEICHERN
5470:   C36E 86 AD              STX   $AD
5480:   C370 20 8E C1           JSR   LOADR     ; LADEN
5482:   C373 A9 05              LDA   #4+1      ; NICHT WARTEN, AN BESTIMMTE
5486:   C375 85 02              STA   ABSFLG    ; ADRESSE LADEN
5490:   C377 A9 00              LDA   #0
5500:   C379 85 B7              STA   $B7       ; KEIN FILENAMEN
5510:   C37B A6 AE              LDX   $AE       ; VARIBLENENDE
5520:   C37D A4 AF              LDY   $AF
5530:   C37F 86 2F              STX   $2F       ; IN VEKTOR
5540:   C381 84 30              STY   $30
5550:   C383 86 AC              STX   $AC       ; UND LADESTARTVEKTOR
5560:   C385 84 AD              STY   $AD
5570:   C387 20 8E C1           JSR   LOADR     ; LADEN
5580:   C38A A6 AE              LDX   $AE       ; ARRAYENDE
5590:   C38C A4 AF              LDY   $AF
5600:   C38E 86 31              STX   $31       ; IN VEKTOR
5610:   C390 84 32              STY   $32
```

```
5620:  C392 A9 04              LDA  #4       ; VON ADRESSE VOM HEADER
5630:  C394 85 02              STA  ABSFLG   ; LADEN
5640:  C396 20 8E C1           JSR  LOADR    ; LADEN
5642:  C399 AD 3C 03           LDA  CASPUF
5642:  C39C 85 33              STA  $33
5644:  C39E AD 3D 03           LDA  CASPUF+1
5644:  C3A1 85 34              STA  $34
5650:  C3A3 20 B7 FF           JSR  HOLSTA   ; STATUS HOLEN
5660:  C3A6 25 BF              AND  $BF      ; EOF-BIT LOESCHEN
5670:  C3A8 F0 05              BEQ  NOFEHL   ; KEIN FEHLER
5680:  C3AA A2 1D              LDX  #$1D     ; OFFSET FUER 'LOAD ERROR'
5690:  C3AC 4C 37 A4           JMP  FEHLAUS   ; FEHLER AUSGEBEN
5700:  C3AF A2 19     NOFEHL   LDX  #$19     ; STRING-DESCRIPTOR
5710:  C3B1 86 16              STX  $16      ; INDEX RUECKSETZEN
5720:  C3B3 A9 00              LDA  #00
5730:  C3B5 85 3A              STA  $3A
5730:  C3B7 85 10              STA  $10      ;CONT SPERREN
5740:  C3B9 60                 RTS
üC000-C3BA
```

```
*************************************************************
 100 REM BASICLOADER FT20
*************************************************************
 110 E=29611:A=28672:PS=0
 120 FOR I=A TO E:READ X:POKE I,X:PS=PS+X:NEXT
 130 IF PS<>114760 THEN PRINT"FEHLER IN DATAS":END
 140 POKE 55,0:POKE 56,112
 150 SYS A:NEW
 160 DATA 169,11,141,8,3,169,112,141,9,3,96,32,115,0,240,
     4,201,95,240,3,76,231
 170 DATA 199,32,115,0,32,32,112,76,174,199,162,0,221,84,
     112,240,8,232,228,4
 180 DATA 208,246,76,8,207,138,10,170,189,90,112,72,189,8
     9,112,72,76,115,0,201
 190 DATA 83,240,7,201,76,240,9,76,8,207,32,115,0,76,3,11
     5,32,115,0,76,83,115
 200 DATA 83,76,86,77,68,98,112,105,113,108,113,91,113,60,
     112,162,5,134,171,162
 210 DATA 0,134,2,32,162,114,165,2,41,2,208,11,162,4,181,
     42,149,171,149,166,202
 220 DATA 208,247,32,183,248,32,40,247,32,252,112,32,11,1
     13,165,185,24,105,1
 230 DATA 202,32,43,113,162,10,185,167,0,32,43,113,162,8,
     200,192,5,234,208,242
 240 DATA 160,0,162,6,177,187,196,183,144,3,169,32,202,32,
     43,113,162,5,200,192
 250 DATA 187,208,237,169,2,133,171,32,11,113,152,32,43,1
     13,132,215,162,9,234
 260 DATA 177,172,32,43,113,162,5,230,172,208,4,230,173,2
     02,202,165,172,197,174
 270 DATA 165,173,229,175,144,231,234,165,215,32,43,113,1
     62,9,136,208,246,200
 280 DATA 132,192,88,24,169,0,141,160,2,76,207,252,32,44,
     200,160,0,132,192,202
 290 DATA 208,253,136,208,250,120,96,160,0,169,2,32,43,11
     3,162,9,136,192,9,208
 300 DATA 244,162,7,198,171,208,238,152,32,43,113,162,9,1
     36,208,247,202,202,96
```

```
310 DATA 133,189,69,215,133,215,169,8,133,163,6,189,173,
    32,145,41,247,32,78
320 DATA 113,162,19,234,9,8,32,78,113,162,16,198,163,208,
    232,96,202,208,253
330 DATA 144,5,162,11,202,208,253,141,32,145,96,165,45,56
    ,233,2,168,165,46,233
340 DATA 0,162,0,240,9,162,0,44,162,1,164,43,165,44,134,
    10,134,147,162,0,134
350 DATA 2,132,195,133,196,32,162,114,32,136,113,76,119,
    225,32,255,113,165,171
360 DATA 201,2,240,8,201,1,208,243,165,185,240,16,169,2,
    5,2,133,2,173,60,3,133
370 DATA 195,173,61,3,133,196,32,244,114,165,2,41,4,208,
    3,32,211,247,32,44,200
380 DATA 164,183,240,11,136,177,187,217,65,3,208,196,152,
    208,245,132,144,32
390 DATA 106,246,173,62,3,56,237,60,3,8,24,101,195,133,1
    74,173,63,3,101,196
400 DATA 40,237,61,3,133,175,32,20,114,165,189,69,215,5,
    144,240,4,169,255,133
410 DATA 144,165,2,208,3,76,65,246,166,45,164,46,96,32,7
    7,114,201,0,240,249
420 DATA 133,171,32,123,114,145,178,200,192,192,208,246,
    240,45,32,77,114,32
430 DATA 123,114,196,147,208,2,145,195,209,195,240,2,134,
    144,69,215,133,215
440 DATA 230,195,208,2,230,196,165,195,197,174,165,196,2
    29,175,144,221,32,123
450 DATA 114,32,252,112,200,132,192,88,24,169,0,141,160,
    2,76,207,252,32,148
460 DATA 248,32,252,112,132,215,169,39,141,40,145,162,1,
    32,142,114,38,189,165
470 DATA 189,201,2,208,245,160,9,32,123,114,201,2,240,24
    9,196,189,208,232,32
480 DATA 123,114,136,208,246,96,169,8,133,163,32,142,114,
    38,189,234,234,234
490 DATA 198,163,208,244,165,189,96,169,2,44,45,145,240,
    251,173,45,145,142,41
500 DATA 145,44,33,145,10,10,10,96,169,0,32,189,255,162,
    1,160,0,32,186,255,32
```

```
510 DATA 3,226,32,84,226,32,3,226,32,253,225,138,168,32,
    186,255,32,3,226,32
520 DATA 230,114,134,172,134,167,132,168,132,173,162,1,1
    34,2,32,3,226,32,230
530 DATA 114,134,174,134,169,132,170,132,175,165,2,9,2,1
    33,2,96,32,11,226,32
540 DATA 138,205,32,247,215,166,20,164,21,96,165,2,41,1,
    240,8,165,172,133,195
550 DATA 165,173,133,196,96,32,38,213,162,5,134,171,32,1
    62,114,169,1,133,185
560 DATA 162,4,181,44,149,171,149,166,202,208,247,32,127,
    112,162,0,134,183,162
570 DATA 5,134,171,202,181,46,149,171,149,166,202,208,247
    ,32,127,112,162,5,134
580 DATA 171,165,51,166,52,133,172,133,167,134,173,134,1
    68,165,55,166,56,133
590 DATA 174,133,169,134,175,134,170,76,127,112,32,162,1
    14,169,1,133,2,165,45
600 DATA 166,46,133,172,134,173,32,136,113,169,5,133,2,1
    69,0,133,183,166,174
610 DATA 164,175,134,47,132,48,134,172,132,173,32,136,11
    3,166,174,164,175,134
620 DATA 49,132,50,169,4,133,2,32,136,113,173,60,3,133,5
    1,173,61,3,133,52,32
630 DATA 183,255,37,191,240,5,162,29,76,55,196,162,25,13
    4,22,169,0,133,58,133
640 DATA 16,96
```

```
************************************************************
 1000 REM LOADER FT64
************************************************************
 1010 E=50105:A=49152:PS=0
 1020 FOR I=A TO E:READ X:POKE I,X:PS=PS+X:NEXT
 1030 IF PS<>120504 THEN PRINT"FEHLER IN DATAS":END
 1040 SYS A
10000 DATA 169,11,141,8,3,169,192,141,9,3,96,32,115,0,240,
      4,201,95,240,3,76,231
10010 DATA 167,32,115,0,32,32,192,76,174,167,162,0,221,84,
      192,240,8,232,228,4
10020 DATA 208,246,76,8,175,138,10,170,189,90,192,72,189,8
      9,192,72,76,115,0,201
10030 DATA 83,240,7,201,76,240,9,76,8,175,32,115,0,76,17,1
      95,32,115,0,76,97,195
10040 DATA 83,76,86,77,68,98,192,111,193,114,193,97,193,60,
      192,162,5,134,171,162
10050 DATA 0,134,2,32,176,194,165,2,41,2,208,11,162,4,181,
      42,149,171,149,166,202
10060 DATA 208,247,32,56,248,32,143,246,32,252,192,32,19,1
      93,165,185,24,105,1
10070 DATA 202,32,51,193,162,8,185,167,0,32,51,193,162,6,2
      00,192,5,234,208,242
10080 DATA 160,0,162,4,177,187,196,183,144,3,169,32,202,32,
      51,193,162,5,200,192
10090 DATA 187,208,237,169,2,133,171,32,19,193,152,32,51,1
      93,132,215,162,7,234
10100 DATA 177,172,32,51,193,162,3,230,172,208,4,230,173,2
      02,202,165,172,197,174
10110 DATA 165,173,229,175,144,231,234,165,215,32,51,193,1
      62,7,136,208,246,200
10120 DATA 132,192,88,24,169,0,141,160,2,76,147,252,32,44,
      168,160,0,132,192,173
10130 DATA 17,208,41,239,141,17,208,202,208,253,136,208,25
      0,120,96,160,0,169,2
10140 DATA 32,51,193,162,7,136,192,9,208,244,162,5,198,171,
      208,238,152,32,51,193
10150 DATA 162,7,136,208,247,202,202,96,133,189,69,215,133
```

```
,215,169,8,133,163,6
10160 DATA 189,165,1,41,247,32,85,193,162,17,234,9,8,32,85,
      193,162,14,198,163
10170 DATA 208,233,96,202,208,253,144,5,162,11,202,208,253
      ,133,1,96,165,45,56
10180 DATA 233,2,168,165,46,233,0,162,0,240,9,162,0,44,162,
      1,164,43,165,44,134
10190 DATA 10,134,147,162,0,134,2,132,195,133,196,32,176,1
      94,32,142,193,76,122
10200 DATA 225,32,10,194,165,171,201,2,240,8,201,1,208,243,
      165,185,240,16,169
10210 DATA 2,5,2,133,2,173,60,3,133,195,173,61,3,133,196,32
      ,2,195,165,2,41,4,208
10220 DATA 8,32,80,247,32,228,255,240,251,32,44,168,164,183
      ,240,11,136,177,187
10230 DATA 217,65,3,208,191,152,208,245,132,144,32,210,245
      ,173,62,3,56,237,60
10240 DATA 3,8,24,101,195,133,174,173,63,3,101,196,40,237,
      61,3,133,175,32,31,194
10250 DATA 165,189,69,215,5,144,240,4,169,255,133,144,165,
      2,208,3,76,169,245,166
10260 DATA 45,164,46,96,32,88,194,201,0,240,249,133,171,32,
      134,194,145,178,200
10270 DATA 192,192,208,246,240,45,32,88,194,32,134,194,196
      ,147,208,2,145,195,209
10280 DATA 195,240,2,134,144,69,215,133,215,230,195,208,2,
      230,196,165,195,197
10290 DATA 174,165,196,229,175,144,221,32,134,194,32,252,1
      92,200,132,192,88,24
10300 DATA 169,0,141,160,2,76,147,252,32,23,248,32,252,192,
      132,215,169,7,141,6
10310 DATA 221,162,1,32,153,194,38,189,165,189,201,2,208,2
      45,160,9,32,134,194
10320 DATA 201,2,240,249,196,189,208,232,32,134,194,136,20
      8,246,96,169,8,133,163
10330 DATA 32,153,194,38,189,234,234,234,198,163,208,244,1
      65,189,96,169,16,44
10340 DATA 13,220,240,251,173,13,221,142,7,221,72,169,25,1
      41,15,221,104,74,74
10350 DATA 96,169,0,32,189,255,162,1,160,0,32,186,255,32,6,
```

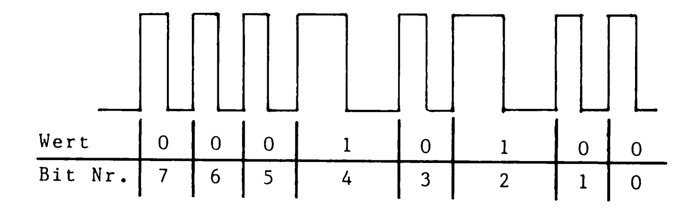
```
       226,32,87,226,32,6
10360 DATA 226,32,0,226,138,168,32,186,255,32,6,226,32,244
      ,194,134,172,134,167
10370 DATA 132,168,132,173,162,1,134,2,32,6,226,32,244,194,
      134,174,134,169,132
10380 DATA 170,132,175,165,2,9,2,133,2,96,32,14,226,32,138
      ,173,32,247,183,166
10390 DATA 20,164,21,96,165,2,41,1,240,8,165,172,133,195,1
      65,173,133,196,96,32
10400 DATA 38,181,162,5,134,171,32,176,194,169,1,133,185,1
      62,4,181,44,149,171
10410 DATA 149,166,202,208,247,32,127,192,162,0,134,183,16
      2,5,134,171,202,181
10420 DATA 46,149,171,149,166,202,208,247,32,127,192,162,5,
      134,171,165,51,166
10430 DATA 52,133,172,133,167,134,173,134,168,165,55,166,56
      ,133,174,133,169,134
10440 DATA 175,134,170,76,127,192,32,176,194,169,1,133,2,1
      65,45,166,46,133,172
10450 DATA 134,173,32,142,193,169,5,133,2,169,0,133,183,16
      6,174,164,175,134,47
10460 DATA 132,48,134,172,132,173,32,142,193,166,174,164,1
      75,134,49,132,50,169
10470 DATA 4,133,2,32,142,193,173,60,3,133,51,173,61,3,133,
      52,32,183,255,37,191
10480 DATA 240,5,162,29,76,55,164,162,25,134,22,169,0,133,
      58,133,16,96
```

12.1 PROGRAMMBESCHREIBUNG

Für interessierte Leser, die zumindest einige Grundkenntnisse in ASSEMBLER-Programmierung haben, möchte ich das FastTape-Programm näher beschreiben. Vielleicht fallen Ihnen noch einige Verbesserungen ein, die Sie in das Programm einbringen wollen. Als Zusatzlektüre empfehle ich Ihnen die entsprechenden Kapitel über die Ein- und Ausgabebausteine in den DATA-Becker Büchern 64 INTERN und VC 20 INTERN.

PRINZIPIELLE ARBEITSWEISE

Einzelne Bits werden durch Rechteckpulse auf Band geschrieben. Das Byte 20, Binär %00010100 sieht folgendermaßen aus:



In folgendem Speicherformat wird ein Programm gespeichert:

1. Synchronisation 1

   5 mal (246 mal Byte 2). Dadurch wird beim Ladevorgang auf den Start eines Bytes synchronisiert.

   Countdown Bytes 9, 8, 7, 6, 5, 4, 3, 2, 1. Hierdurch wird das Ende der Synchronisation gekennzeichnet.

2. Vorspann, bestehend aus

   Sekundäradresse-1
   Start- und Endadresse
   Filenamen

   Der Vorspann wird immer auf 192 Zeichen mit "SPACE" aufgefüllt und beim Laden im Cassetten-Puffer abgelegt. Somit ist es auch hier möglich, mit dem Filenamen Maschinenprogramme zu übergeben, wie es im Kapitel 7.3 beschrieben wurde.

3. Synchronisation 2

   Wie Synchronisation 1, aber nur 2 mal.

4. Null-Byte Kennzeichnung dafür, daß jetzt die Daten folgen.

5. Daten

6. Prüfsumme, 255 mal

Im Unterprogramm INIT wird der Vektor zur Interpreterschleife auf das FastTape-Programm umgelenkt. Dadurch wird erreicht, daß jedesmal, wenn ein Befehl ausgewertet werden soll, FastTape angesprungen wird. Im Unterprogramm START wird nun getestet, ob das erste Zeichen der FastTape-Code (Pfeil nach links) ist. Ist das Zeichen kein FastTape-Code, wird in die Interpreterschleife zurückgesprungen.

Wird der FastTape-Code gefunden, wird über das Unterprogramm SUCH durch Vergleich mit der Buchstaben- tabelle TAB die entsprechende Startadresse bestimmt. Wird der Buchstabe nicht gefunden, wird ein"?SYNTAX ERROR" ausgegeben.

Sonst wird die entsprechende Befehlsstartadresse auf den STACK geschoben. Dadurch verzweigt das Programm nach dem RTS

der CHRGET-Routine zu der entsprechenden Adresse plus eins und kehrt nach Ausführung zu der Adresse $C01D($701D) zurück.

Bevor ich die einzelnen Hauptprogramme erkläre, möchte ich einige häufig benutzte Unterprogramme beschreiben.

Byte auf Band schreiben : WBYTE $C133

Die Übergabe des zu schreibenden Bytes erfolgt über den Akkumulator. Dieses Byte wird in die Speicherstelle $BD geschrieben und dann mit der EXOR-Prüfsumme durch EXKLUSIV ODER verknüpft.

In $C139 wird 8 (ein Byte = acht Bit) in den Bitcounter geschrieben. Mit SHIFT beginnt die Schleife, die acht Bit auf Band schreibt. Zuerst wird das höchstwertige Bit in das Carry-Flag geschoben. Das PORT-Byte wird nun in den Akku geladen und das Schreibbyte (Bit drei) gelöscht.

Entsprechend der Zeitschleife T1LOOP (bei gesetzten Carry-Flag zusätzlich T2LOOP) ist die Schreibleitung nun "high". Am Ende der Zeitschleife wird der Akkuinhalt in den Port übertragen mit der Folge, daß die Schreibleitung "low" wird.

In $C149 wird dann im Akku wieder Bit drei gesetzt und erneut in die Zeitschleife T1LOOP verzweigt. Am Ende wird dann durch die Übertragung des Akkuinhalts in den Port die Schreibleitung wieder "high".

In $C150 wird der Bitcounter dekrementiert und getestet. Wurden alle acht Bits übertragen, wird zurückgesprungen.

Byte vom Band lesen :   HOLBYT $C286

Zuerst wird ab $C286 der Bitcounter auf 8 gesetzt. Dann werden acht Bit nacheinander über HOLBIT eingeladen und über das Carry-Flag in $BD geschoben. Am Ende ($C269) wird über den Akku das eingelesene Byte übergeben.

Bit vom Band lesen :   HOLBIT $C299

Die Schleife $C299 - $C29E wartet bis Bit vier im ICR2 des CIA2 "high" wird. Dieses Bit wird gesetzt, wenn an der READ Leitung (D/4) des Cassettenports ein Signal anliegt. Dann wird das ICR1 geladen und das X-Register als Highbyte in den Zähler B vom CIA2 geschrieben und ICR1 auf den STACK gerettet.

In $C2A9 wird Bit 1,3 und 4 im Kontrollregister B gesetzt, was zur Folge hat, daß Zähler B einmal abwärts zählt und gestartet wird. Danach wird der gerettete ICR1 Inhalt zurückgeholt und durch zweimaliges Rechtsschiften Bit eins ins Carry-Flag geschoben. Dieses Bit wird dann "high", wenn der Zähler bis auf 0 gezählt hat, bevor er neu gestartet wurde.

Auf diese Weise wird festgestellt, ob es lange dauerte, bis die Leseleitung wieder high war, was einem gesetzten Bit entspricht.

Parameter holen :   GETPARA $C2BO

Diese Routine holt die mit den Befehlen übergebenen Parameter und schreibt sie in die entsprechenden Speicherstellen. Weiterhin wird entsprechend der Parameter das Flag ABSFLG gesetzt.

Dieses Flag ist ein Binärflag und steuert folgende Funktionen:

Bit	Wert	Wirkung
0	1	1 = File an eine beim L-Befehl übermittelte Adresse laden
1	2	1 = File absolut laden oder speichern
2	4	1 = Beim C64 nicht warten, sondern sofort laden

Synchronisation schreiben :SYNCH $C113

Dieses Unterprogramm schreibt sooft, wie in $AB angegeben ist, 246 mal das Byte 2 auf Band. Zuletzt wird ein Countdown aus den Bytes 9, 8, 7, 6, 5, 4, 3, 2, 1 auf Band geschrieben.

Synchronisation suchen : SCOUNT $C258

Zuerst wird die Recordertaste abgefragt. Dann wird der Motor gestartet, Bildschirm abgeschaltet (nur C-64) und das Prüfsummenbyte auf 0 gesetzt. Ab $C260 wird 7 in das Lowbyte geschrieben und das X-Register mit für das Highbyte des Timers mit 1 geladen

Ab SSTART werden dann solange Bits eingelesen, bis der Bytewert der eingelesenen Bits genau 2 wird. Dann werden alle folgenden Zweier-Bytes der Synchronisation eingelesen und festgestellt, ob danach ein Countdown ab 9 folgt.

Wird ein fehlerfreier Countdown gefunden, wird zum Hauptprogramm zurückgesprungen.

Vorspann einlesen : SSYNCH $C20A

Dieses Unterprogramm sucht mit SCOUNT die nächste Synchronisation. Mit Hilfe des letzten von SCOUNT eingelesenen Bytes stellt es fest, ob es sich um Synchronisation 1 oder 2 handelt. Ist die gefundene Synchronisation nicht Synchronisation 1, wird weitergesucht. Sonst wird das letzte Byte als Sekundäradresse in $AB gerettet. Daraufhin wird der Vorspann eingelesen und im Cassettenpuffer abgelegt. Danach wird über die Betriebssystemroutine ab $FC93 der Motor ausgeschaltet und beim C-64 der Bildschirm wieder eingeschaltet.

Programm laden : PLOAD $C21F

Durch dieses Unterprogramm wird das Programm eingelesen. Zuerst wird mit SCOUNT die Synchronisation 2 gesucht und damit auch die Leseroutine auf das Band synchronisiert.

LLOOP ist die eigentliche Laderoutine. Mit HOLBYT wird ein Byte eingelesen und über den Startvektor indirekt-indiziert in den Speicher geschrieben und verglichen. Bei V (VERIFY) wird nur verglichen. Tritt dabei ein Fehler auf, wird das Statusbyte gesetzt. Danach wird die EXOR-Prüfsumme gebildet und der Startvektor inkrementiert.

Die Schleife LLOOP wird solange durchlaufen, bis die Differenz zwischen Start- und Endvektor gleich null ist. Dann wird noch die abgespeicherte Prüfsumme eingelesen, der Motor aus- und beim C64 der Bildschirm angeschaltet. Über die Betriebssystem-Routine $FC93 wird zum Hauptprogramm zurückgesprungen.

Als letztes möchte ich die Hauptroutinen etwas näher beschreiben.

SAVE ROUTINE $C063

In $AB wird zunächst die Wiederholungszahl der Synchronisation 1 abgelegt und das Flag ABSFLG gelöscht. Danach werden über GETPARA die einzelnen Fileparameter eingelesen. Entsprechend ABSFLG werden dann die Basicvektoren in die Lade-, Start- und Endevektoren übertragen.

Ab $C07F wird die Recordertaste abgefragt, "SAVING name" ausgegeben, der Motor gestartet und beim C-64 der Bildschirm ausgeschaltet. Über das Unterprogramm SYNCH wird nun die Synchronisation auf Band geschrieben. Danach wird die um eins erhöhte Sekundäradresse auf Band übertragen. Darauf folgend wird ab $C094 der Vorspann, bestehend aus Start- und Endadresse, Filename und Füllbytes auf Band geschrieben. Zum Schluß folgt die Synchronisation 2, die zur Kennzeichnung mit einem Null-Byte abgeschlossen ist.

Ab $C0CB (PRGLOOP) folgt die eigentliche Programmspeicherroutine. Das Programm wird über den Programmstartvektor $AC/$AD indirekt - indiziert aus dem Speicher gelesen und an die Datassette übertragen. Danach wird der Vektor $AC/AD inkrementiert und mit dem Programmendvektor verglichen.

Diese Schleife wird solange durchlaufen, bis der Startvektor gleich dem Endvektor ist. Als Abschluß wird dann 255 mal die EXOR-Prüfsumme auf Band geschrieben. Nun wird 1 in das Motorflag geschrieben, um das Abschalten des Motors vorzubereiten, und das Speichern über die Betriebssystemroutine in $FC93 beendet.

LADEROUTINE $C18E

Als erstes wird eine Synchronisation durch das Unterprogramm SSYNCH gesucht. Ist das letzte Byte der gefundenen Synchronisation, das in $AB geschrieben wurde, null, handelt es sich um eine Synchronisation 2, und es wird wieder zum Start dieser Routine gesprungen. Ist $AB ungleich null, wurde eine Synchronisation 1 gefunden, und der Wert in $AB ist die Sekundäradresse.

Ist das File mit der Sekundäradresse eins abgespeichert, oder ist die beim Ladebefehl übermittelte Sekundäradresse gleich eins, wird in ABSFLG Bit eins gesetzt, und die Startadresse aus dem Cassettenpuffer wird in den Ladestartvektor geschrieben. Danach wird ab RELLOD "Found name" ausgegeben.

Das Unterprogramm ADTEST testet nun anhand von Bit null von ABSFLG, ob anstatt der im Cassettenpuffer stehenden Startadresse eine andere durch den L-Befehl übermittelte Adresse als Ladestartvektor benutzt werden soll. Ist Bit null von ABSFLG gesetzt, wird die mit L übermittelte Adresse, die in $AC/$AD steht, in den Startvektor geschrieben.

Entsprechend Bit zwei von ABSFLG wartet der Computer nun auf einen Tastendruck von Ihnen oder nicht. Nach dem Test, ob die STOP-Taste gedrückt wurde, wird der Filename des gefundenen Files mit dem des gewünschten verglichen, falls ein Name übergeben wurde. Stimmen die Namen überein, kann der Ladevorgang beginnen, und das Statusbyte wird gelöscht.

Durch die Addition der Programmlänge, gebildet aus der Differenz der Start- und Endadresse aus dem Vorspann, und dem Startvektor wird die Endadresse berechnet und in den Endvektor geschrieben. Das Unterprogramm PLOAD lädt dann das Programm.

Danach werden die Prüfsummen verglichen und ggf. der Status gesetzt. Wenn in ABSFLG kein Bit gesetzt ist, werden über

die Betriebssytemroutine STOEND die Basic-Vektoren auf das Programmende gesetzt.

Die Routine MERGE, LOAD und VERIFY laufen alle über die LOAD Routine.

Bei MERGE wird der Startvektor entsprechend dem Basic-Programmendvektor gesetzt. Danach wird relativ geladen. Bei VERIFY wird das VERIFY-Flag über das X-Register auf eins, bei LOAD auf null gesetzt.

Vor der LOAD-Routine werden über GETPARA noch die weiteren Parameter eingelesen und in ABSFLG der entsprechende Wert gesetzt.

DATENSPEICHERUNG UND DATEN LESEN

Die Daten werden als 3 Blöcke abgespeichert.

1. Block : Variablen
   Bereich zwischen den Zeigern 45, 46/47, 48

2. Block : Felder
   Bereich zwischen den Zeigern 47, 48/49, 50

3. Block : Strings
   Bereich zwischen den Zeigern 51, 52/55, 56

Die Länge der einzelnen Blöcke wird im Vorspann durch die Start- und Endadresse übergeben.

DATASAVE $C311

Ehe die Daten abgespeichert werden können, müssen sie auf minimalen Platz komprimiert werden. Deshalb wird zu Beginn die Garbage-Collection aufgerufen.

Mit Hilfe von GETPARA wird nun der Filename gesetzt. Danach wird als Sekundäradresse 1 gegeben, da ein bestimmter Speicherbereich abgespeichert werden soll. Durch ULOOP werden die Zeiger auf Anfangs- bzw. Endadressse des Variablenbereichs in die Start- und Endvektoren übertragen.

Zuletzt wird dieser Bereich durch das Unterprogramm ABSOLUT absolut abgespeichert.

Die Felder und Strings werden danach analog zu den Variablen ohne Filenamen über ABSOLUT abgespeichert.

DATALOAD $C361

Zuerst wird über GETPARA der Filename gesetzt. Durch Setzen des Bit null in ABSFLG wird erreicht, daß der Bereich an eine bestimmte Adresse geladen wird und die Adressen im Vorspann ignoriert werden.

Als Ladeadresse wird die Variablen-Startadresse in den Startvektor übertragen. Über LOADR wird nun der Variablenbereich eingelesen. Danach wird zusaätzlich Bit zwei in ABSFLG gesetzt, damit der Rechner nach dem Finden der nächsten Synchronisation 1 nicht auf einen Tastendruck des Bedieners wartet. Weiterhin wird kein Filename gegeben. Der Imjalt von $AE, $AF setzt den Zeiger auf das Ende des beschriebenen Bereichs. Dieser Zeiger wird zugleich als Startvektor für die Felder in den Ladestartvektor übernommen, damit bei erneutem Aufruf von LOADR die Felder eingelesen werden. Danach wird $AE, $AF in den Zeiger auf das Ende der Felder geschrieben.

Da die Stringtabelle vom Basic-RAM-Ende nach unten wächst, wird deren Ende nicht durch die Basic-Programmlänge beeinflußt. Aus diesem Grunde wird dieser Bereich immer wieder dahingeladen, von wo er abgespeichert wurde. Darum wird Bit null in ABSFLG gelöscht. Danach wird die Tabelle durch die Routine LOADR geladen.

Der Zeiger auf das untere Ende der Stringtabelle wird dann aus dem Cassettenpuffer gelesen und in 51, 52 gespeichert.

Zu guter letzt wird der Status getestet und ggfs. eine Fehlermeldung ausgegeben. Ist kein Fehler aufgetreten, wird der Stringdescriptor zurückgesetzt und zum Hauptprogramm zurückgesprungen.

13. DATENVERARBEITUNG MIT FASTTAPE

In diesem Kapitel möchte ich Ihnen zeigen, daß auch mit einem Cassettenrecorder mit der geeigneten Software eine effiziente Datenverarbeitung möglich ist.

Das in diesem Kapitel abgedruckte Datenverarbeitungsprogramm hat folgende Spezifikationen:

- Daten speichern und laden mit FastTape. Je nach Größe der Datei dauert der Speicher- und Ladevorgang zwischen 20 Sekunden und 1,5 Minuten.

- Die Wahl der Datensatzanzahl ist Ihnen überlassen. Das Maximum ist von der freien Speicherkapazität des Rechners abhängig.

- Ein Datensatz kann in beliebig viele Felder unterteilt werden.

- Ein Feld darf maximal 80 Zeichen haben.

- Die Bezeichnung der einzelnen Felder ist frei wählbar und kann auch nach der ersten Festlegung geändert werden.

- Gesucht werden kann in einzelnen oder mehreren Feldern. Dabei können Sie wählen, ob ein oder alle Suchkriterien erfüllt werden sollen.

- Die Suchgeschwindigkeit ist in allen Feldern gleich.

- Es kann nach Übereinstimmung, Ungleichheit, größer und kleiner gesucht werden.

  Falls Sie eine Befehlerweiterung mit einem INSTRING-Befehl besitzen, kann auch nach Teilbereichen innerhalb von Feldern gesucht werden.

- Wenn in mehreren Feldern gesucht wird, kann für jedes Feld eine andere Suchfunktion bestimmt werden.

- Es kann jederzeit eine HELP-SEITE angezeigt werden, die die einzelnen Befehle erklärt.

- Sie können mit gefundenen Daten Listen erstellen, die Sie formatiert ausgeben können.

- Sie können frei bestimmen, welche Felder ausgegeben werden sollen.

- Die einzelnen Felder können mit oder ohne Feldbezeichnung ausgegeben werden.

- Listen können nach einem beliebigen Feld sortiert werden.

Ich habe dieses Programm so geschrieben, daß es ohne Änderungen sowohl auf dem VC 20 als auch auf dem C-64 läuft. Diese allgemeinen Anpassungen brauchen aber einen gewissen Speicherplatz und können bei Bedarf geändert bzw. gelöscht werden, sodaß wieder mehr Speicherplatz zur Verfügung steht. Damit Sie bei Bedarf das Programm an Ihre eigenen Bedürfnisse anpassen können, finden Sie am Ende dieses Kapitels eine detaillierte Programmbeschreibung.

Doch kommen wir nun zur Bedienung des Programms. Es ist möglich, mit diesem Programm eine Datei beliebiger Art einzurichten und zu bearbeiten. Sie können ein Literatur- oder Rezeptverzeichnis anlegen, Ihre Plattensammlung erfassen, oder die Adressen Ihrer Freunde verwalten.

Um Ihnen die Funktion und Bedienung dieses Programms darzulegen, ist es am besten, wenn Sie, geleitet durch diese Bedienungsanleitung, eine Übungsdatei anlegen. Da ein Rezeptverzeichnis oder ähnliches zu diesem Zwecke zu umfangreich ist, legen Sie am besten eine Adressdatei an.

Ich gebe zu, daß dies schon eine äußerst abgedroschene Dateiart ist, doch ist es mit einer Adressdatei möglich, mit wenig Schreibarbeit eine Datei mit mehreren Datensätzen zu schreiben.

Nach dem Starten mit RUN erscheint auf Ihrem Bildschirm das Menue.

```
* * * M E N U E * * *

  Datei pflegen
  blaettern
  Liste ausgeben
  Liste sortieren
  Daten eingeben
  Datei laden
  Datei speichern
  Datei einrichten
  Feldbez. aendern
```

Da Sie noch keine Datei angelegt haben, wählen Sie als erstes Punkt 9 des Menues.

DATEI EINRICHTEN

In diesem Menuepunkt müssen Sie die Grundcharakteristika Ihrer Datei eingeben. Sie werden gebeten, die Feldanzahl und die maximale Datensatzanzahl einzugeben. Diese Werte können nicht mehr geändert werden.

Danach werden Sie aufgefordert, Feldbezeichnungen für die einzelnen Felder einzugeben. Für unsere Übungsadressdatei

geben Sie bitte als maximale Datensatzanzahl 10 oder mehr ein und als Feldanzahl 7. Die einzelnen Felder bezeichnen Sie am besten folgendermaßen:

1. Vorname
2. Name
3. Straße und Hausnummer
4. PLZ
5. Wohnort
6. Telefon
7. Hobby

Wenn Sie diese Eingaben getätigt haben, kehren Sie wieder zum Hauptmenue zurück. Ein nochmaliges Anwählen dieses Menuepunktes ist nun nicht mehr möglich.

Da Sie das Programm ja noch nicht richtig ausgetestet haben, es können sich immer einmal Fehler beim Abtippen einschleichen, speichern Sie diese Datei vorsichtshalber schon mal ohne Daten ab. Wählen Sie dafür den Punkt 7 "Datei speichern" an.

Als erstes werden Sie nach dem Filenamen gefragt. Geben Sie "Adressen" ein und drücken Sie RETURN. Nun werden Sie gebeten, eine Datencassette einzulegen. Legen Sie also eine Cassette als Datencassette in Ihren Recorder. Achten Sie dabei darauf, daß Sie die Cassette nicht ganz am Anfang stehen haben, damit die Datei auch sicher gespeichert wird. Drücken Sie RETURN und folgen der Aufforderung "PRESS RECORD & PLAY ON TAPE".

Nachdem alle Variablen gespeichert wurden, kehrt der Rechner zum Menue zurück. Nun wollen wir einige Datensätze eingeben.

DATEN EINGEBEN

Nachdem Sie diesen Menuepunkt angewählt haben, erscheint folgendes Bild:

```
* DATEN EINGEBEN *

es existieren 0 Datensaetze


Vorname :

Name :

Strasse u. Hausnummer :

Plz :

Wohnort :

Telefon :

Hobby :
```

Der Cursor steht unter der Feldbezeichnung "Vorname". Sie können nun in jedes Feld bis zu 80 Zeichen schreiben. Solange Sie innerhalb eines Feldes bleiben, steht Ihnen der volle Commodore-Bildschirmeditor zur Verfügung. Füllen Sie die Felder einmal folgendermaßen aus:

```
* DATEN EINGEBEN *

es existieren 0 Datensaetze


Vorname :
Max
Name :
Meier
Strasse u. Hausnummer :
Oberstr. 3
Plz :
4000
Wohnort :
Duesseldorf
Telefon :

Hobby :
Computer
```

Nachdem Sie die Eingabe im letzten Feld mit RETURN abgeschlossen haben, erscheint "RETURN" in der letzten Bildschirmzeile.

Sie haben nun folgende Eingabemöglichkeiten:

RETURN - Der Datensatz wird in die Datei übernommen, und Sie können den nächsten Datensatz eingeben.

Q - Der Datensatz wird nicht übernommen, und das Programm kehrt zum Menue zurück.

K - Der Cursor erscheint wieder im ersten Feld, und Sie können die Eingaben korrigieren.

beliebige andere Taste - Der Datensatz wird übernommen, und das Programm kehrt zum Menue zurück.

Geben Sie also bei fehlerfreier Eingabe RETURN. Im Kopf wird nun angezeigt, daß ein Datensatz existiert und der Cursor erscheint wieder im ersten Eingabefeld. In den Feldern steht nun noch die letzte Eingabe, die Sie jetzt ggf. überschreiben können. Falls sich einige Felder nicht ändern, können Sie sie durch Drücken von RETURN diesen Datensatz übernehmen. Geben Sie auf die oben beschriebene Weise einmal vier Adressen ein.

Falls eine Erweiterung mit "INSTRING"-Befehl zur Verfügung steht, geben Sie auch einmal den nächsten Datensatz ein:

```
* DATEN EINGEBEN *

es existieren 4 Datensaetze


Vorname :
Marlies
Name :
Bange
Strasse u. Hausnummer :
Kaiserstr. 44
PLZ :
4150
Wohnort :
Krefeld
Telefon :
344322
Hobby :
Kochen/Computer
```

Kehren Sie nun durch Drücken einer beliebigen Taste zum Menue zurück.

BLÄTTERN

Wählen Sie nun Menuepunkt 2. In diesem Modus können Sie sich die Daten in der Reihenfolge in der Sie sie eingegeben haben ansehen und ggf. ändern.

Durch Betätigen der RETURN-Taste können Sie jeweils um einen Datensatz weiterblättern. Mit der "-"-Taste können Sie zurückblättern. Wenn Sie die "K"-Taste drücken, erscheint der Cursor im ersten Feld, und Sie können den angezeigten Datensatz ändern. Mit der "Q"-Taste kommen Sie wieder in das Menue.

```
* * BLAETTERN * *

Datensatz Nr. 2


Vorname :
Peter
Name :
Namenlos
Strasse u. Hausnummer :
Dorfstr. 123
PLZ :
1234
Wohnort :
Nirgendwo
Telefon :
123456
Hobby :
Computer
```

DATEN PFLEGEN

Unter Menuepunkt 1 gelangen Sie in diesen Programmteil. Sie können hier die Daten selektieren, löschen, ändern und Listen erstellen.

Als erstes suchen wir im Feld "Name" alle Datensätze, in denen der Name mit "M" beginnt. Geben Sie also auf die Frage, nach welchen Feldern gesucht werden soll, eine 2 ein. Daraufhin erscheint die entsprechende Feldbezeichnung invers auf dem Bildschirm. In der vorletzten Bildschirmzeile erscheint:

[illegible]

Das sind die Auswahlmöglichkeiten, die Sie haben.

1. Teilstring (nur mit INSTRING-Befehl)
Sie suchen nach einer Zeichenfolge, die an beliebiger Stelle innerhalb des Feldes stehen kann.

2. Gleich
Sie suchen nach einer signifikanten Zeichenfolge, die genau gleich Ihrem Suchstring ist. Es werden dabei immer nur soviele Zeichen verglichen, wie Sie als Suchstring eingeben. Geben Sie z.B. "Ma" ein, findet das Programm Mann, Maus, Mama etc.. Dies gilt analog auch bei den folgenden Möglichkeiten.

3. Ungleich
Hier werden die Strings gefunden, die nicht mit dem Suchstring übereinstimmen.

4. Größer oder gleich
Hier findet das Programm alle Strings, die größer oder gleich sind. Geben Sie z.B. M bei dem Namen ein, werden alle Namen gefunden, die mit M oder einem folgenden Buchstaben (N, O, P, Q...) beginnen.

5. Kleiner oder gleich
Geben Sie als Suchstring M ein, werden alle Namen, die mit A - M beginnen, gefunden.

Drücken Sie "2", und Sie erhalten folgendes Bild:

```
SUCHEN / AENDERN
1. Vorname
2. Name
3. Strasse u. Hausnummer
4. PLZ
5. Wohnort
6. Telefon
7. Hobby

Nach welchen Feldern soll gesucht werden


 = Name
```

Wenn Sie innerhalb weiterer Felder suchen wollen, müssen Sie nur die entsprechende Zahl eingeben.

Wir wollen aber jetzt erst einmal nur nach dem Namen suchen. Durch Drücken der RETURN-Taste schließen Sie die Eingabe ab und werden nun gefragt, ob ein Datensatz bei einer oder allen Übereinstimmungen gefunden werden soll. Da Sie ja sowieso nur ein Auswahlkriterium eingegeben haben, geben Sie RETURN.

Es erscheint nun die bestimmte Feldbezeichnung und der Cursor. Da wir alle Namen suchen wollen, die mit "M" beginnen, geben Sie M ein und RETURN. Daraufhin erscheint in der letzten Bildschirmzeile wieder "RETURN". Wenn Sie nun "K" drücken, können Sie die Eingabe korrigieren. Drücken Sie RETURN, beginnt der Rechner mit der Suche und zeigt dann die Trefferzahl, die Datensatznummer des angezeigten Datensatzes und den Datensatz selber an.

```
              SUCHEN / AENDERN

 3  Treffer
Datensatz Nr. 0


Vorname :
Max
Name :
Meier
Strasse u. Hausnummer :
Oberstr. 3
PLZ :
4000
Wohnort :
Duesseldorf
Telefon :

Hobby :
Computer



    << RETURN >>
```

Sie können nun wie unter "Blättern" in den gefundenen Datensätzen blättern.

Weiterhin können Sie sie durch "K" korrigieren und durch "L" löschen. Mit "Q" gelangen Sie wieder in das Menue.

Machen Sie sich jetzt am besten etwas mit den Menuepunkten 1 und 2 vertraut, damit Sie sehen, wie und wie schnell die gewünschten Datensätze gesucht und korrigiert werden können. Suchen Sie dann einmal nach einem Kriterium, das auf mehrere Datensätze zutrifft.

Wenn Sie so eine Liste aus mehreren Datensätzen erstellt haben, können Sie sie über Menuepunkt 4 nach einem beliebigen Feld, z.B. "Namen", sortieren.

Über den Menuepunkt 3 können Sie die Liste auf dem Bildschirm oder auf einem Drucker ausgeben.

LISTE AUSGEBEN

In diesem Menuepunkt werden Sie gefragt, welche Felder ausgedruckt werden sollen. Durch die Eingabe der Feldindice können Sie die gewünschten Felder wählen und mit RETURN die Eingabe beenden.

Wenn Sie stattdessen "K" eingeben, werden die angezeigten Felder gelöscht, und Sie können die Eingabe wiederholen. Nachdem Sie die gewünschten Felder gewählt haben, drücken Sie RETURN.

Als nächstes werden Sie nach einer Überschrift gefragt. Diese wird dann oberhalb Ihrer Liste ausgegeben. Bei der Druckerausgabe wird die Überschrift in Breitschrift ausgedruckt.

Danach können Sie wählen, ob die Feldbezeichnungen mit ausgedruckt werden sollen.

Als letztes wird danach gefragt, ob die Ausgabe auf den Bildschirm oder über einen Drucker erfolgen soll. Bei der Druckerausgabe könnten Sie dann folgendes Bild erhalten:

```
Computer-Freaks

Marlies
Anmut
Kaiserstr. 12
4150
Krefeld
344356
Kochen/Computer


Max
Meier
Oberstr. 3
4000
Duesseldorf

Computer
```

```
Peter
Namenlos
Dorfstr. 123
1234
Nirgendwo
123456
Computer
```

Das Programm ist so konzipiert, daß jedes Feld in eine eigene Zeile geschrieben wird. Durch Programmänderungen in den Zeilen 2870 - 2930 können Sie das Ausdruckformat entsprechend Ihren Wünschen ändern.

Wenn Sie die Liste auf den Bildschirm ausgeben, können Sie die Ausgabe durch Drücken der "SPACE" - Taste anhalten und wieder starten.

Ich hoffe, daß Ihnen die Bedienung dieses Programms nach einigem Üben keine Schwierigkeiten mehr macht.

```
   1 goto 1060
************************************************************
  90 rem "    Such-Routine"
************************************************************
 100 if(f>fe)or(n>=as)then return
 110 a=sf(f)
 120 on a gosub 150,190,250,280,220
 130 f=f+1:goto 100
 140 : rem "  Teilbereich
 150 t=instr(u$(f),s$(f))
 160 if t<>0 then n=n+1
 170 return
 180 : rem "  ganz '='
 190 if s$(f)=left$(u$(f),len(s$(f)))then n=n+1
 200 return
 210 : rem "  ganz '<='
 220 if s$(f)>=left$(u$(f),len(s$(f)))then n=n+1
 230 return
 240 : rem "  ganz '<>'
 250 if s$(f)<>left$(u$(f),len(s$(f)))then n=n+1
 260 return
 270 : rem "  ganz '>='
 280 if s$(f)<=left$(u$(f),len(s$(f)))then n=n+1
 290 return
************************************************************
 1000 rem "Warteschleife
************************************************************
 1010 poke 198,0:wait 198,1:get a$:a=asc(a$)
      :rem "Warteschleife
 1020 if a=133 then gosub 3750
 1030 return
************************************************************
 1050 rem"  Initialisierung
************************************************************
 1060 dw$="[home,down25]"
 1070 sys 65517:sp=peek(781):ze=peek(782)
      :rem " Bildschirmdaten holen
 1080 c1=53281:c2=1:if sp=22 then c1=36879:c2=8
```

```
1090 dim z$(4):z$(0)=" T":z$(1)=" =":z$(2)="<>":z$(3)=">="
     :z$(4)="<="
************************************************************
1100 rem "      Hauptprogramm
************************************************************
1110 print chr$(14):poke c1,6*c2
1120 gosub 3560:print:print:ta=int(sp/5)-2
1130 print tab(ta)"[white,rvs]1[rvoff,space]Datei pflegen"
1140 print tab(ta)"[rvs]2[rvoff,space]blaettern "
1150 print tab(ta)"[rvs]3[rvoff,space]Liste ausgeben"
1160 print tab(ta)"[rvs]4[rvoff,space]Liste sortieren"
1170 print tab(ta)"[rvs]5[rvoff,space]Daten eingeben "
1180 print tab(ta)"[rvs]6[rvoff,space]Datei laden  "
1190 print tab(ta)"[rvs]7[rvoff,space]Datei speichern"
1200 print tab(ta)"[rvs]8[rvoff,space]Datei einrichten"
1210 print tab(ta)"[rvs]9[rvoff,space]Feldbez. aendern"
1220 print tab(ta)"[rvs]E[rvoff,space4,rvs]ENDE[rvoff]"
1230 gosub 1010:a=val(a$):if a$="e"then 1270
1250 on a gosub 2130,2000,2700,2990,1860,1570,1460,1700,1
     780
1260 goto 1110
************************************************************
1265 rem "Programm ENDEN
************************************************************
1270 if(fl and 2)=2 or(fl and 4)=0 then end
1280 print"[clear,down20]":print tab(sp/2-8)"[white]trotz
     dem ENDEN":gosub 1310
1290 if fl<>255 then 1110
1300 end
************************************************************
1305 rem "         FEHLER
************************************************************
1310 poke c1,5*c2
1320 print"[home]":print"[down7]"
1330 print:print tab(sp/2-3)"[red,rvs]DATEI"
1340 print:print tab(sp/2-6)"[rvs]WURDE NICHT"
1350 print:print tab(sp/2-7)"[rvs]ABGESPEICHERT"
1360 for i=0 to 15:get a$:if a$<>""then i=15:next
     :goto 1440
1370 next
```

```
 1380 print"[home]":print"[down7]"
 1390 print:print tab(sp/2-3)"[red]DATEI"
 1400 print:print tab(sp/2-6)"WURDE NICHT"
 1410 print:print tab(sp/2-7)"ABGESPEICHERT"
 1420 for i=0 to 15:get a$:if a$=""then i=15:next:goto 1310
 1430 next
 1440 if a$="j"then fl=255
 1450 return
************************************************************
 1460 rem "         Speichern
************************************************************
 1470 if(fl and 1)=0 then return
 1480 poke c1,7*c2:print"[blue]":gosub 3590
 1490 print:print:print"  "f$"[up]":input"Filename";f$
 1500 print:print"Datencassette einlegen"
 1510 print:print:print"   << RETURN >>"
 1520 gosub 1010
 1530 if a$="k"then print"[home]":goto 1490
 1540 if a$="q"then return
 1550 if a<>13 then 1520
 1560 _dsf$:fl=fl or 2:return
************************************************************
 1570 rem "          Laden
************************************************************
 1580 if(fl and 6)=4 then gosub 1680:if fl<>255 then return
 1590 poke c1,7*c2:print"[blue]":gosub 3600
 1600 print:print:print"  "f$"[up]":input"Filename";f$
 1610 print:print"Datencassette einlegen"
 1620 print:print:print"   << RETURN >>"
 1630 gosub 1010
 1640 if a$="k"then print"[home]":goto 1600
 1650 if a$="q"then return
 1660 if a<>13 then 1630
 1670 _dlf$:fl=fl and 251:return
 1680 print"[clear,down20]":print tab(sp/2-8)"[white]trotz
      dem LADEN"
 1690 gosub 1310:return
************************************************************
 1700 rem "     Datei erstellen
************************************************************
```

```
 1710 poke c1,8*c2:if fl then return
 1720 gosub 3550:rem " Titel schreiben
 1730 print:print:input"Feldanzahl";fa
 1740 print:input"[down]max Datensaetze";ra
 1750 print:print"<< RETURN >> ":gosub 1010
      :if a<>13 then print"[home]":goto 1730
 1760 dim m$(fa),e$(fa,ra),sf(fa),s$(fa),u$(fa),gf%(ra)
 1770 fl=1:rem "  Flag setzen
************************************************************
 1775 rem "    Feldbezeichnungen eingeben
************************************************************
 1780 if(fl and 1)=0 then return
 1790 gosub 3550:rem "Ueberschrift schreiben
 1800 print:print"Geben Sie die ";:if sp<40 then print
      :print
 1810 print"Feldbezeichnungen ein"
 1820 print:for i=0 to fa-1:print:print"[rvs]Feld "i+1"[rv
        off]"
 1830   print"  "m$(i)"[up]":input m$(i):next
 1840 print left$(dw$,ze)"<< RETURN >>";:gosub 1010
      :if a$="k"then print"[home]":goto 1800
 1850 fl=fl or 4:return
************************************************************
 1860 rem "     Daten eingeben
************************************************************
 1865 if fl and 1=0 then return
 1870 poke c1,3*c2:print"[blue]"
 1880 close 1:open 1,0
 1890 gosub 3580:print:print"es existieren"rt;
      :if sp<40 then print
 1900 print"Datensaetze"
 1910 r1=rt:r2=rt:if(rt>0)and(a$<>"k")then r1=rt-1
 1920 gosub 3650:gosub 3670:rem "  Felder anzeigen
 1930 print left$(dw$,ze)"[green,space3]<< RETURN >> [whit
      e]";:gosub 1010
 1940 if a$="q"then return
 1950 if a$="k"then 1890
 1960 if a<>13 then fl=fl or 12:rt=rt+1:return
 1970 rt=rt+1:if rt<ra then 1890
 1980 gosub 3580:print:print:print"maximale Satzzahl ";
```

```
      :if sp<40 then print
 1990 print:print"erreicht":gosub 1010:return
***********************************************************
 2000 rem "       Blaettern
***********************************************************
 2005 if(fl and 8)=0 then return
 2010 poke c1,6*c2:print"[white]"
 2020 r1=0
 2025 if r1<0 then r1=r1+rt
 2030 gosub 3610:print:print"Datensatz Nr."r1
 2040 gosub 3650
 2050 print left$(dw$,ze)"[green,space3]<< RETURN >> [whit
      e]";:gosub 1010
 2060 if a$="-"then r1=r1-1:goto 2025
 2070 if a$="q"then return
 2080 if a$="a"then gosub 2560:goto 2060
 2090 if a<>13 then 2050
 2100 if r1<rt-1 then r1=r1+1:goto 2030
 2110 gosub 3610:print"[down8,right,rvs,space]Keine weiter
      n Daten "
 2120 r1=0:goto 2050
***********************************************************
 2130 rem "     Datei pflegen
***********************************************************
 2135 if(fl and 8)=0 then return
 2140 poke c1,4*c2
 2150 for i=0 to fa-1:sf(i)=0:next:as=0
 2160 gosub 3620:for i=0 to fa-1:print i+1"[left]. "m$(i)
      :next
 2170 print:print"Nach welchen Feldern ";
      :if sp<40 then print:print
 2180 print"soll gesucht werden":print:print
 2190 gosub 1010:if a=13 then 2262
 2200 if a$="q"then return
 2210 if a$="k"then 2150
 2220 i=val(a$):if i<1 or i>fa then 2190
 2230 if sf(i-1)then 2190
 2240 d6=peek(214):rem " Cursorzeile retten
 2250 sf(i-1)=1:print tab(3)"[rvs]"m$(i-1):gosub 3710
 2260 print left$(dw$,d6+1)z$(sf(i-1)-1):as=as+1:goto 2190
```

```
2262 gosub 3620:print:print"Finden bei"
2264 print:print tab(4)"1 = einer"
2265 print:print tab(4)"0 = allen"
2266 print:input"Uebereinstimmungen";ue
2267 if ue then as=ue
2270 gosub 3620:close 1:open 1,0
2280 print:print"Geben Sie Suchstrings ";
     :if sp<40 then print:print
2290 print"ein":print:print
2300 for f=0 to fa-1:if sf(f)=0 then 2330
2310   print"[rvs]"m$(f)" : "
2320   print s$(f)"[up]":input#1,s$(f):print
2330   next
2340 print left$(dw$,ze)"<< RETURN >>";:gosub 1010
     :if a$="k"then print"[home]":goto 2280
2350 if a$="q"then return
2360 if a<>13 goto 2340
************************************************************
2370 rem "          Suchen
************************************************************
2372 for f=0 to fa-1:if sf(f)then fb=f:f=fa
2374   next:rem "niedrigstes und hoechstes
2376 for f=fa-1 to 0 step-1:if sf(f)then fe=f:f=0
2378   next:rem "Suchfeld feststellen
2380 z=0:for r =0 to rt-1
2390   for f=fb to fe-1:if sf(f)then u$(f)=e$(f,r)
2400     next:rem "Untersuchstrings umspeichern
2410   n=0:f=fb:gosub 100:if n>=as then gf%(z)=r:z=z+1
2420   next:if z=0 then 2130
************************************************************
2430 rem "     Datensaetze anzeigen
************************************************************
2440 i=0
2450 gosub 3620:print:print z" Treffer"
     :print"Datensatz Nr."gf%(i)
2460 r1=gf%(i):gosub 3650
2470 print left$(dw$,ze)"[green,space3]<< RETURN >> [whit
     e]";:gosub 1010
2480 if a$="-"and i>0 then i=i-1:goto 2450
2490 if a$="q"then return
```

```
2500 if a$="a"then gosub 2560:goto 2480
2510 if a$="l"then 2640
2520 if a<>13 then 2470
2530 if i<z-1 then i=i+1:goto 2450
2540 gosub 3620:print"[down8,right3,rvs,space]Keine weite
     rn Daten "
2550 print left$(dw$,ze)"[green,space3]<< RETURN >> [whit
     e]";:gosub 1010:goto 2450
************************************************************
2560 rem "          Aendern
************************************************************
2570 close 1:open 1,0:r2=r1:gosub 3670
2580 print left$(dw$,ze)"[green,space3]<< RETURN >> [whit
     e]";:gosub 1010
2600 if a$="k"then 2570
2610 fl=fl or 4:fl=fl and 253:return
2620 return
************************************************************
2630 rem "         Loeschen
************************************************************
2640 for i1=gf%(i)+1 to rt-1:rem " Daten
2650   for j=0 to fa-1:rem   "       umspeichern
2660     e$(j,i1-1)=e$(j,i1)
2670     next j,i1
2680 z=z-1:rt=rt-1:fl=fl or 4:goto 2450
************************************************************
2690 rem "      Liste ausgeben
************************************************************
2695 if z=0 then return
2700 poke c1,7*c2:print"[blue]"
2710 for i=0 to fa-1:sf(i)=0:next
2720 gosub 3630
2725 for i=0 to fa-1:print i+1"[left]. "m$(i):next
2730 print:print"Welche Felder sollen ";
     :if sp<40 then print:print
2740 print"ausgedruckt werden":print:print
2750 gosub 1010:if a=13 then 2810
2760 if a$="q"then return
2770 if a$="k"then 2710
2780 i=val(a$):if i<1 or i>fa then 2750
```

```
 2790 if sf(i-1)then 2750
 2800 sf(i-1)=1:print"[rvs]"m$(i-1):print:goto 2750
 2810 print:input"Ueberschrift";ub$
 2820 print:print"Feldbezeichnungen":print
      :input"ausdrucken [1/0]";fm$
 2830 fm=val(fm$)
 2840 print:input"Drucker=4/Bilds.=0";dv
 2850 if dv then open 4,dv,7:cmd 4
 2860 print chr$(14)ub$chr$(15):print:print
 2870 for r=0 to z-1
 2880   for f=0 to fa-1:if sf(f)=0 then 2910
 2890     if fm then print:print"[rvs]"m$(f)" : [rvoff]"
 2900     print e$(f,gf%(r))
 2910     next:if dv then 2930
 2920   get a$:if a$<>""then poke 198,0:wait 198,1:get a$
 2930   print:print:next
 2940 if dv then print#4:close 4:return
 2950 print:print"    << RETURN >>"
 2960 gosub 1010:if a<>13 then 2960
 2970 return
************************************************************
 2980 rem "         Sortieren
************************************************************
 2990 if z=0 then return
 3000 gosub 3640
 3010 for i=0 to fa-1:print i+1"[left]. "m$(i):next
 3020 print"[down]Nach welchem Feld ";:if sp<40 then print
      :print
 3030 input"soll sortiert werden";sf:sf=sf-1
 3040 f1=0:fh=0:fh=peek(49)+peek(50)*256
      :rem "Obergrenze retten
 3050 f1=peek(47)+peek(48)*256:rem " Untergrenze retten
 3060 dim n$(z),z(z+1)
 3070 q=1:for i=0 to z-1:rem "    Sortierfeld
 3080   n$(q)=e$(sf,gf%(i)):rem " umspeichern
 3090   z(q)=q:q=q+1:next
 3100 gosub 3240:rem "         Sortierroutine
 3110 for x=0 to fa-1
 3130   for y=0 to z-1
 3140     n$(y+1)=e$(x,gf%(y))
```

```
 3160      next
 3170    for y=0 to z-1
 3180      e$(x,gf%(y))=n$(z(y+1))
 3200      next y,x
 3210 for i=0 to z-1:n$(i)="":next:rem "Strings von n$(
      ) loeschen
 3220 gosub 3520:rem "Wiederherstellung der alten Zeiger
 3230 return
************************************************************
 3235 rem "    Sortierroutine
************************************************************
 3240 ti$="000000":print"[clear]"
 3250 l=int(z/2)+1
 3260 r=z:print"[clear]"r"[left,space3]"
 3270 if l>1 then 3340
 3280 if r<=1 then 3330
 3290 h2$=n$(l):h1=z(l)
 3300 n$(l)=n$(r):z(l)=z(r)
 3310 n$(r)=h2$:z(r)=h1
 3320 r=r-1
 3330 goto 3350
 3340 l=l-1
 3350 j=l
 3360 i=2*j
 3370 h2$=n$(j):h1=z(j)
 3380 if i>r then 3480
 3390 if i>=r then 3420
 3400 if n$(i)>=n$(i+1)then 3420
 3410 i=i+1:print"[clear]"i"[left,space3]"
 3420 if i>r then 3480
 3430 if h2$>=n$(i)then 3480
 3440 n$(j)=n$(i):z(j)=z(i)
 3450 j=i
 3460 i=2*j
 3470 goto 3390
 3480 n$(j)=h2$:z(j)=h1
 3490 if r<>1 then 3270
 3500 print"[clear,down2]Sortierzeit";ti$
 3510 return
************************************************************
```

```
3515 rem "Pointer wiederherstellen
***********************************************************
3520 j=peek(47)+peek(48)*256+(fh-f1)
3530 poke 49,j and 255:poke 50,int(j/256)
3540 return
***********************************************************
3545 rem "    Ueberschriften
***********************************************************
3550 print"[clear]";:print tab(sp/2-10)"* DATEI ERSTELLEN
      *":return
3560 print"[clear]";:print tab(sp/2-10)"[white]* * * M E
     N U E * * *":return
3570 print"[clear]";:print tab(sp/2-9)"* DATEI PFLEGEN *"
     :return
3580 print"[clear,blue]";:print tab(sp/2-10)"* DATEN EING
     EBEN *":return
3590 print"[clear]";:print tab(sp/2-8)"* DATEN SPEICHERN
     *":return
3600 print"[clear]";:print tab(sp/2-6)"* DATEN LADEN *"
     :return
3610 print"[clear]";:print tab(sp/2-8)"* * BLAETTERN * *"
     :return
3620 print"[clear]";:print tab(sp/2-8)"SUCHEN / AENDERN"
     :return
3630 print"[clear]";:print tab(sp/2-7)"LISTE AUSGEBEN"
     :return
3640 print"[clear]";:print tab(sp/2-7)"LISTE SORTIEREN"
     :return
***********************************************************
3645 rem "    Felder ausgeben
***********************************************************
3650 print"[home,down5]":for f=0 to fa-1
       :print"[rvs]"m$(f)" : "
3660   print e$(f,r1):next:return
3670 print"[home,down5]":for f=0 to fa-1
       :print"[rvs]"m$(f)" : "
3680   print e$(f,r1)"[up]":a=len(e$(f,r1))
3690   if a>sp then for x=0 to a/sp:print"[up]";:next
3700   input#1,e$(f,r2):print:next:return
***********************************************************
```

```
3705 rem "   Suchform waehlen
************************************************************
3710 print left$(dw$,ze-1)"[rvs]1:T/2:=/3:<>/4:>/5
     :<[rvoff]";
3720 gosub 1010:j=val(a$):if j<1 or j>5 then 3720
3730 sf(i-1)=j
3740 print left$(dw$,ze-1)"                    ";:return
************************************************************
3750 rem "      Helpseite
************************************************************
3760 print"[clear]* * * * * * H E L P S E I T E * * * * *"
3770 print:print"Wenn  << RETURN >> erscheint haben Sie"
3780 print:print"Optionen :":print:print
3790 print"RETURN  = weitermachen":print
3800 print"K       = letzte Eingabe wiederholen":print
3810 print"Q       = Programmpunkt abbrechen":print
3820 print"-       = einen Satz zurueck":print
3830 print"a       = einen Satz aendern":print
3840 print"l       = einen Satz loeschen":print
3850 get a$:if a$=""then 3850
3860 return
5000 "**********************************
5010 "*  Dateiverwaltung mit Fasttape  *
5020 "*       (c) Dirk Paulissen       *
5030 "*                                *
5040 "* Verwendete Variablen :         *
5050 "* fa     = Feldanzahl            *
5060 "* ra     = Recordanzahl(max)     *
5070 "* rt     =   - '' - (aktuell)    *
5080 "* f      = aktuelles Feld        *
5090 "* r      = aktueller Record      *
5100 "* sp     = Spaltenanzahl         *
5110 "* ze     = Zeilenzahl            *
5120 "* a$     = Fragestring           *
5130 "* fl     = Flag                  *
5140 "* e$(f,r) = Datenstring          *
5150 "* sf(f)   = Suchfeldflag         *
5160 "*         1 : Teilstring suchen  *
5170 "*         2 : ganz '='           *
5180 "*         3 : ganz '<>'          *
```

```
5190 "*          4 : ganz '>='          *
5200 "*          5 : ganz '<='          *
5220 "* s$(f)    = Suchstring           *
5230 "* U$(f)    = Untersuchstring      *
5240 "* gf%()    = Nummer der gefunde-  *
5250 "*            nen Datensaetze      *
5300 "**********************************
```

13.1 PROGRAMMBESCHREIBUNG

Der Programmaufbau ist folgender:

90 - 1030	Schnelle Unterprogramme
1050 - 1090	Initialisierung
1100 - 1260	Hauptprogramm (Menue)
1270 - 3540	Unterprogramme, die vom Menue aufgerufen werden
3550 - 3640	Überschriften
3650 - 3850	Unterprogramme, die von anderen Unterprogrammen aufgerufen werden
5000 - Ende	Copyright Vermerk und Aufzählung der benutzten Variablen

ARBEITSWEISE

Die Anpassung an die verschiedenen Computer geschieht in Zeile 1070 durch den Aufruf der Betriebssystemroutine SCREEN (65517). Diese Routine legt die Spalten- und Zeilenzahl in den Speicherstellen 781 und 782 ab. Entsprechend dieser Werte werden die Speicherstellen und die Werte für die FarbPOKE - Befehle gesetzt (C1, C2).

Das Hauptprogramm besteht nur aus dem Menue und der Verzweigung in das gewählte Unterprogramm. Zur Datensicherheit wurde ein Flag fl eingeführt, welches immer am Anfang eines Unterprogramms getestet und innerhalb der Unterprogramme gesetzt wird. Die einzelnen Bits von fl haben folgende Bedeutung, wenn sie gesetzt sind:

Bit	Wert	Bedeutung
0	1	Datei eingerichtet
1	2	Datei gespeichert
2	4	Datei verändert
3	8	Datei enthält Datensätze

Wird versucht, eine neue Datei zu laden oder das Programm zu beenden, ehe eine geänderte Datei abgespeichert wurde, wird in die Fehlerroutine 1305 - 1450 verzweigt.

Die Programmteile "Laden" und "Speichern" verstehen sich von selbst.

DATEI ERSTELLEN (1700 - 1850)

Nachdem der Titel durch Zeile 1720 geschrieben wurde, wird per INPUT die Feldanzahl (fa) und Datensatzanzahl (ra) eingelesen. Durch die Abfrage in 1750 wird es ermöglicht, eine fehlerhafte Eingabe zu korrigieren.

Entsprechend fa und ra werden in 1760 die Felder dimensioniert und in 1770 Bit null in fl gesetzt.

In Zeile 1780 wird Bit null in fl getestet, da die Routine zur Eingabe und Änderung der Feldbezeichnungen auch vom Menue angesprungen werden kann.

In Zeile 1800 und 1810 wird der auszugebene String entsprechend der Spaltenzahl des Computers in einer oder zwei Zeilen ausgedruckt.

Innerhalb einer Schleife (Zeile 1820 - 1830) werden die einzelnen Feldbezeichnungen in m$(fa) eingelesen. Eventuell vorhandene werden auf dem Bildschirm ausgegeben und können durch RETURN übernommen werden.

Durch die Abfrage in Zeile 1840 ist wieder eine Korrektur möglich.

DATEN EINGEBEN (1860 - 1990)

In Zeile 1865 wird fl daraufhin getestet, ob schon eine Datei erstellt wurde.

In der Zeile 1880 wird durch OPEN 1,0 die Tastatur eröffnet. Das hat zur Folge, daß bei einem folgenden INPUT#1 kein Fragezeichen ausgegeben wird. Aus Sicherheitsgründen wird vor jedem OPEN - Befehl die entsprechende Datei geschlossen.

Über die Zeilen 1890 - 1900 wird die Überschrift und entsprechend der Spaltenzahl ein String ausgegeben. In den Unterprogrammen, welche die einzelnen Felder eines Satzes anzeigen (3650 - 3660) bzw. neue Felder einlesen (3670 - 3700), wird r1 bzw. r2 als Satzzeiger benutzt. Aus diesem Grunde wird in Zeile 1910 der Zeiger auf den einzugebenden Satz rt in r1 und r2 übertragen. Wurde schon ein Satz eingegeben (rt größer null), wird r1 dekrementiert, sodaß die Felder des zuletzt eingegebenen Satzes angezeigt werden. Ab Zeile 1930 wird abgefragt, ob abgebrochen (a$ = "q"), korrigiert (a$ = "k"), beendet (a ungleich 13) oder weitergemacht werden soll. Ist a = 13, wird in Zeile 1970 getestet, ob die maximale Satzzahl erreicht wurde.

BLÄTTERN (2000)

Mit Hilfe der Laufvariablen r1 werden über das Unterprogramm "Felder ausgeben" die einzelnen Datensätze angezeigt und dann die Tastatur über die "Warteschleife" (1010) abgefragt. Entsprechend der Eingabe wird weitergeblättert (2100), zum Menue zurückgekehrt (2070) oder in das Unterprogramm "Ändern" (2080) verzweigt.

DATEI PFLEGEN (2130)

Nach Abfrage von fl und BildschirmPOKE werden die Suchflags (sf(i)) auf null gesetzt (2150). Über die "Warteschleife" wird ab Zeile 2190 abgefragt, ab welchem Feld gesucht werden soll. Wurde eine gültige, noch nicht gewählte Feldnummer eingegeben (der Test dafür ist in Zeile 2220 - 2230), wird in Zeile 2240 die aktuelle Cursorzeile in d6 gerettet und in Zeile 2250 die Feldbezeichnung des gewünschten Feldes ausgedruckt.

Über das Unterprogramm "Suchform wählen" ab Zeile 3710 wird der Index der gewünschten Suchform in das entsprechende Suchflag eingelesen.

Mit Hilfe der geretteten Cursorzeile (d6) wird dann in Zeile 2260 das entsprechende Suchformzeichen vor die Feldbezeichnung gesetzt und zur Abfrage nach weiteren Suchfeldern zurückgesprungen.

In den Zeilen 2262 - 2267 wird die gewünschte Anzahl der Übereinstimmungen in ue geschrieben. Über die Zeilen 2280 - 2360 werden die Suchstrings in das Feld s$(f) eingelesen.

Da die eigentliche Suchroutine in jedem Datensatz die Felder von einem Startfeld bis zu einem Endfeld untersucht, wird in Zeile 2372 - 2378 der niedrigste und höchste Feldindex festgestellt und in fb, fe gerettet, um so die Suchzeit zu optimieren.

Die Hauptsuchschleife über die einzelnen Datensätze beginnt ab Zeile 2380. In Zeile 2390 werden die zu untersuchenden Strings in das Feld u$(i) übertragen. In der Zeile 2410 wird der Index für gefundene Übereinstimmungen n auf null und f auf das Startfeld gesetzt, bevor in die eigentliche Suchroutine ab Zeile 100 gesprungen wird.

Die Suchroutine liegt aus Zeitgründen am Programmanfang, da die Zieladresse nach einem Sprung (GOTO, GOSUB) entweder hinter der aufrufenden Zeile oder ab Programmanfang gesucht wird. Würde die Routine am Programmende liegen, müßte bei einem Sprung in eine vorherige Zeile immer das ganze Programm nach der Zielzeilennummer durchsucht werden, was bei längeren Dateien die Suchzeit merklich verlängern würde.

Am Anfang der Suchroutine wird getestet, ob schon alle gewünschten Felder durchsucht wurden, oder ob die gewünschte Anzahl der Übereinstimmungen gefunden wurde. Ist dies nicht der Fall, wird über das Feld sf(f) der gewünschte Vergleich

angesprungen.

Ist der entsprechende Vergleich logisch wahr, wird der Übereinstimmungszähler erhöht. Nachdem der Feldindex f erhöht wurde, wird zum Start zurückgesprungen. Nach dem Rücksprung in die Zeile 2410 wird, wenn die gewünschte Übereinstimmungszahl erreicht wurde, die Datensatznummer in das Feld gf%(z) geschrieben und die Trefferzahl z inkrementiert.

Nach Abschluß des Suchens werden die Treffer analog zum Unterprogramm "Blättern" angezeigt (2430 - 2550).

LISTE AUSGEBEN (2690 - 2970)

Diese Routine gibt die durch das Feld gf%(i) bestimmten Datensätze entweder auf den Bildschirm oder dem Drucker aus. Die Abfrage nach den auszudruckenden Feldern ist analog zu der Feldbestimmung im Unterprogramm "Suchen".

SORTIEREN (2980 - 3510)

Nach der Abfrage, ob eine Liste erstellt wurde und nach der Ausgabe der Überschrift auf den Bildschirm (2990 - 3000) wird das Feld, nach dem sortiert werden soll, abgefragt, und der entsprechende Index in sf geschrieben.

Für die folgende Sortierroutine werden einige Hilfsfelder benötigt, deren Größe vom Listenumfang abhängt. Da Felder aber normalerweise nur einmal dimensioniert werden dürfen, hätte ich bei der Initialisierung die Hilfsfelder n$(i) und z(i) mit der maximalen Datensatzanzahl dimensionieren müssen. Das würde einerseits viel Speicherplatz schlucken und andererseits die Lade- und Speicherzeit vom bzw. auf Band verlängern. Mit einem Kunstgriff ist es aber möglich, Fehler selektiv zu löschen und somit redimensionierbar zu machen.

Dieser Kunstgriff besteht darin, vor einer neuen Dimensionierung die Ober- und Untergrenze des Bereiches, in dem der Computer die Felder ablegt zu retten. Die danach dimensionierten Felder können dann durch Rücksetzen der Zeiger auf die alte Länge der Feldertabelle gelöscht werden.

Aus diesem Grunde werden in den Zeilen 3040 und 3050 die Zeiger auf den Start und das Ende der Feldertabelle in f1 und fh gerettet.

In der Zeile 3060 werden die Hilfsfelder entsprechend des Listenumfangs dimensioniert.

Durch die Schleife in den Zeilen 3070 bis 3090 wird das entsprechende Datensatzfeld eines jeden in der Liste befindlichen Datensatzes in n$(i) übertragen. Dieses Stringfeld wird später sortiert. Damit danach die entsprechenden anderen Felder eines Datensatzes dem sortierten Feld zugeordnet werden können, wird in z(i) ein "Pointer" angelegt, der die Position des entsprechenden Datensatzes innerhalb der Liste enthält.

Die Sortierroutine verarbeitet nur Felder mit Indizes ab eins. Im Datenfeld e$(f,r) sind die Datenfelder aber ab Index null abgespeichert. Aus diesem Grund müssen die Daten und der Pointer jeweils in eine Feldvariable mit einem Index, der um eins höher ist als der im Datenfeld, umgespeichert werden.

Bei der Umspeicherroutine wird der erhöhte Index dreimal benötigt (Index von n$, Index von z, Pointer in z). Damit innerhalb der Umspeicherschleife nicht dreimal "i+1" berechnet werden muß, habe ich aus Zeitgründen eine Hilfsvariable q definiert, die innerhalb der Schleife nur einmal inkrementiert werden muß.

In Zeile 3100 wird in die Sortierroutine (3240 - 3510) verzweigt. Diese Routine sortiert das eindimensionale Feld n$(i) durch entsprechendes Vertauschen der Inhalte von n$(i). Bei jedem Tauschvorgang wird auch der Inhalt des

Pointers vertauscht. Am besten wird dies durch ein Beispiel klar. Nehmen wir an, fünf Namen sollen sortiert werden, die in folgender Reihenfolge in den Feldern stehen:

```
n$(1) = Max      z(1) = 1
n$(2) = Alfred   z(2) = 2
n$(3) = Peter    z(3) = 3
n$(4) = Dirk     z(4) = 4
```

Nach dem.Sortieren erhalten wir:

```
n$(1) = Alfred   z(1) = 2
n$(2) = Dirk     z(2) = 4
n$(3) = Max      z(3) = 1
n$(4) = Peter    z(4) = 3
```

Der Index von z gibt nun die aufsteigende Reihenfolge der sortierten Strings an und der Inhalt von jedem z die Datensatznummer, die an der entsprechenden Position stehen muß.

In der Schleife Zeile 3110 - 3200 werden die Datenfelder entsprechend dem Pointer vertauscht.

Die äußere Schleife zählt über die Felder. In der ersten inneren Schleife wird das entsprechende Feld (durch x bestimmt) aller Datensätze in der alten Reihenfolge in das Feld n$(i) übertragen.

In der zweiten inneren Schleife werden dann die Inhalte von n$(i) entsprechend dem Pointer z(i) in das Datenfeld sortiert zurückgeschrieben.

Ab Zeile 3210 werden die Hilfsfelder wieder gelöscht. Zuerst werden die Strings gelöscht, um den Stringvektor im Merkheft des Rechners zurückzusetzen. Über das Unterprogramm in den Zeilen 3520 - 3540 wird zu dem aktuellen Start der Feldertabelle die gerettete Länge addiert und als

Feldertabellenende in den Vektor 49, 50 geschrieben.

Ich hoffe, daß Ihnen nun die Arbeitsweise dieses Programms einigermaßen klargeworden ist und Sie nun wissen, wo und wie Sie dieses Programm Ihren eigenen Wünschen noch besser anpassen können. Ich könnte mir z.B. noch folgende Anpassungen denken:

- Druckerausgabe; der Datei entsprechend formatierte Ausgabe.

- Helpseite; diese Bildschirmausgabe könnte man auf einer separaten Bildschirmseite schreiben, die dann mit Hilfe von POKE-Befehlen mit der aktuellen Bildschirmseite ausgetauscht wird. Das ist eine Anpassung, die besonders für C-64 Anwender interessant ist; hier könnte man die zusätzliche Bildschirmseite unter das Betriebssystem-ROM (ab $A000) schreiben, sodaß kein Basic-Speicherplatz verlorengeht.

14. CC-INHALT UND KATALOG FÜR FASTTAPE-CASSETTEN

Im Kapitel 7 habe ich Ihnen ein Programm beschrieben, das ein Inhaltsverzeichnis Ihrer Cassetten erstellt. Solch eine Inhaltsübersicht möchten Sie sich bestimmt auch für Ihre FastTape-Cassetten anlegen. Darum finden Sie hier ein Listing eines Programms, welches genauso arbeitet wie das in Kapitel 7.

```
 90 REM***********************************
 95 REM*                                 *
100 REM*     INHALTSVERZEICHNIS          *
101 REM*                                 *
103 REM*   VON FASTTAPE CASSETTEN        *
104 REM*                                 *
105 REM*     (C) DIRK PAULISSEN          *
106 REM*                                 *
107 REM***********************************
108 REM V=.7:KL=.54 :REM KONSTANTEN FUER ZAEHLERBERECHNU
    NGEN C60
109 V=.5:KL=.482:REM KONSTANTEN FUER ZAEHLERBERECHNUNGEN
     C90
110 PA=828:REM STARTADRESSE IM CASSETTENPUFFER
115 SA=171 :REM SEKUNDAERADRESSE +1
116 PRINT"[CLEAR,SPACE11]INHALTSVERZEICHNIS"
117 PRINT:PRINT"          FUER FASTTAPE CASSETTEN"
120 PRINT"[DOWN2]DRUCKEN ? [J/N]  ";D$
130 GET D$:IF D$=""THEN 130
136 TA$=CHR$(16)
140 T1$=TA$+"05":T2$=TA$+"10":T3$=TA$+"19":T4$=TA$+"27"
    :T5$=TA$+"35"
142 T6$=TA$+"43":T7$=TA$+"61":REM TAB'S
145 IF D$<>"J"THEN 170
150 OPEN 2,4 :REM DRUCKER OEFFNEN
160 PRINT#2,T1$"LFN";T2$"ZAEHLER";T3$"K-BYTE";T4$" ANFAN
    G"T5$"   ENDE";
```

```
162 PRINT#2,T6$"       NAME"T7$"SEKADR."
165 PRINT#2
190 SYS 49674:REM VC20 = 29183
200 A$=":"+RIGHT$("       "+STR$(PEEK(PA)+(256*PEEK(PA+1))
    -1),6)
210 B$="C"+RIGHT$("       "+STR$(PEEK(PA+2)+(256*PEEK(PA+3)
    )-1),6)
220 C$=": ":FOR I=5 TO 20:C$=C$+CHR$(PEEK(PA+I)):NEXT
    :C$=C$+":"
230 T=VAL(RIGHT$(B$,5))-VAL(RIGHT$(A$,5))
    :K$=STR$(INT(T/1024*1000+.5)/1000)
235 K$=RIGHT$("       "+STR$(INT(T/1024*1000+.5)/1000),6)
240 SA$=STR$(PEEK(SA)-1)
250 PRINT Z;K$A$B$:PRINT C$:PRINT
252 FA=1+2**(Z/4400):REM ABHAENGIGER FAKTOR ZUR ZAEHLER
    BERECHNUNG
255 Z1=Z:Z=INT(Z+V*FA+(T/1024)*(KL*FA))
    :REM BANDZAEHLER BERECHNEN
256 PRINT"NACHSTER ZAEHLERSTAND CA."Z
260 N=N+1:N$=RIGHT$("       "+STR$(N),3)
270 IF D$="J"THEN PRINT#2,T1$;N$;T2$;Z1;T3$;K$;T4$;A$;T5$
    ;B$;T6$;C$;T7$SA$
300 GOTO 190
```

Das in Kapitel 10.1 beschriebene Katalogprogramm arbeitet nach entsprechender Änderung ebenfalls mit FastTape.

Sie müssen nur die normalen LOAD-Befehle gegen FastTape-Befehle austauschen. Da die Synchronisation bei FastTape-Programmen bedeutend kürzer ist als bei normal abgespeicherten Programmen, empfiehlt es sich, die Programme nicht direkt hintereinander abzuspeichern, sondern jeweils ca. 5 sec Zwischenraum zu lassen.

15. BACKUP VON DISK AUF CC UND UMGEKEHRT

Dieses Kapitel ist für diejenigen bestimmt, die sowohl ein Diskettengerät besitzen als auch einen Datenrecorder. Eine Datencassette ist bedeutend robuster und weniger störungsanfällig als eine Diskette. Darum lohnt es sich auch für einen Diskettenbesitzer, wichtige Programme auf Cassette zu speichern. Auch gibt es viele, die selbst nur einen Cassettenrecorder haben, aber einen Freund besitzen, der eine Diskettenstation hat. Mit den folgenden Programmen ist es nun viel einfacher, Programme untereinander auszutauschen.

15.1 BACKUP VON CASSETTE AUF DISK

Mit diesem Backup-Programm haben Sie die Möglichkeit, mehrere FastTape-Programme, die hintereinander auf einer Cassette gespeichert sind, auf eine Diskette umzuspeichern.

Für das Programm ist es unerheblich, ob es sich um Basic- oder Maschinensprache-Programme handelt. Die Programme werden alle so kopiert, daß sie voll lauffähig sind, egal von welchem Speichermedium Sie es wieder einladen. Wichtig ist nur, daß alle Programme, die Sie kopieren wollen, einen Namen haben, da die Diskettenstation im Gegensatz zum Datenrecorder namenlose Files nicht akzeptiert.

Da das Backup-Programm mit FastTape arbeitet, müssen Sie das FastTape-Programm laden, bevor Sie das Backup-Programm starten.

Die Bedienung ist denkbar einfach.

Geben Sie das Programm in Ihren Rechner ein und speichern Sie es vor dem Starten ab, da es sich selber nach dem

Starten löscht. Wenn Sie nun das Programm starten, wird zuerst das Maschinenprogramm in den Speicher ab $C3C0 ($73C0) hinter das FastTape-Programm geschrieben. Dann werden Sie nach der Anzahl der zu kopierenden Programme gefragt. Dieser Wert wird in der Speicherstelle 767 für das Maschinenprogramm zwischengespeichert.

Als nächstes müssen Sie eine formatierte Diskette in das Diskettenlaufwerk und die Cassette mit den zu kopierenden Programmen in den Datenrecorder legen. Nach Drücken der RETURN-Taste verzweigt das Basic-Programm in das Maschinenprogramm. Dies liest nun das erste Programm in den Basicram (der komplette Basicram wird als Puffer benutzt). Danach wird das Programm aus dem Speicher auf Diskette geschrieben. Dieser Vorgang wird nun der Anzahl der Programme entsprechend wiederholt. Am Ende kehrt der Rechner wieder in den Direktmodus zurück.

Wollen Sie nun noch weitere Programme kopieren, müssen Sie die Programmanzahl in die Speicherstelle 767 POKEn und das Maschinenprogramm erneut durch SYS 50112 (SYS 28672 beim VC 20) starten.

```
85:     C3C0                    .PAG 61
86:     C3C0                    .TIT "BACKUP T=>D"
90:     C3C0                    .OPT P2,O1
100:    C3C0                    *=   $C3C0       ;VC20 = $73C0
110:    C3C0
110:    C3C0
115:                  ; *********************************
120:                  ; *                               *
130:                  ; *   BACKUP VON CC AUF DIKETTE   *
135:                  ; *                               *
140:                  ; *         MIT FASTTAPE          *
145:                  ; *                               *
146:                  ; *********************************
150:    C3C0
150:    C3C0
150:    C3C0
150:    C3C0
155:                  ; AUSDRUECKE IN KLAMMERN GELTEN
157:                  ;         FUR    VC 20
160:    C3C0
160:    C3C0
200:    000D          CR       =    13       ;
205:    0002          ABSFLG   =    2
210:    00AC          STARTV   =    $AC
220:    00AE          ENDVEC   =    $AE
230:    033C          CASPUF   =    $33C
240:    02FF          ANZAHL   =    $2FF
245:    0803          CSTART   =    $803     ; ($1203) PUFFERSTART
250:    C18E          LOADR    =    $C18E    ; ($7188) FASTTAPE LADEN
260:    A642          NEW      =    $A642    ; ($C642) BASIC-BEF. NEW
270:    A437          ERROR    =    $A437    ; ($C437) BASIC-WARMSTART
280:    F693          MSSG     =    $F693    ; ($F72C) MELDUNG AUSGEBEN
290:    C3C0
290:    C3C0
300:    FFB7          STATUS   =    $FFB7
310:    FFBA          SETLFS   =    $FFBA
320:    FFBD          SETNAM   =    $FFBD
325:    FFC0          OPEN     =    $FFC0
330:    FFC3          CLOSE    =    $FFC3
335:    FFC9          CHKOUT   =    $FFC9
340:    FFCC          CLRCH    =    $FFCC
350:    FFD2          BSOUT    =    $FFD2
950:    C3C0
950:    C3C0
955:    C3C0
955:    C3C0
960:                  ; HAUPTPROGRAMM
970:    C3C0
970:    C3C0
1000:   C3C0 A9 00    START    LDA  #$00
1010:   C3C2 20 42 A6          JSR  NEW
1020:   C3C5 20 70 C4          JSR  CLOAD
1030:   C3C8 20 D4 C3          JSR  DSAVE
1040:   C3CB CE FF 02          DEC  ANZAHL
1050:   C3CE F0 03             BEQ  ENDE
1060:   C3D0 4C C0 C3          JMP  START
1070:   C3D3 60       ENDE     RTS
1082:   C3D4
1082:   C3D4
```

```
1084:  C3D4
1084:  C3D4
1086:               ; AUF DISK SPEICHERN
1088:  C3D4
1088:  C3D4
1089:  C3D4 A9 10   DSAVE    LDA  #$10
1090:  C3D6 A2 41            LDX  #$41      ; NAMENPARAMETER
1092:  C3D8 A0 03            LDY  #$03      ; SETZEN
1093:  C3DA 20 BD FF         JSR  SETNAM
1095:  C3DD 20 93 F6         JSR  MSSG      ; ' SAVING NAME ' AUSGEBEN
1100:  C3E0 A9 0D            LDA  #CR
1110:  C3E2 20 D2 FF         JSR  BSOUT     ; NEUE ZEILE
1120:  C3E5 A2 10            LDX  #$10
1130:  C3E7 CA      ELOOP    DEX
1140:  C3E8 BD 41 03         LDA  CASPUF+5,X
1150:  C3EB C9 20            CMP  #$20
1160:  C3ED D0 04            BNE  ENDNAM
1170:  C3EF E0 00            CPX  #$00
1180:  C3F1 D0 F4            BNE  ELOOP
1190:  C3F3 E8      ENDNAM   INX            ; ENDE DES FILENAMEN
1200:  C3F4 A9 2C            LDA  #","      ; SUCHEN UND ' ,P,W '
1210:  C3F6 9D 41 03         STA  CASPUF+5,X
1220:  C3F9 E8               INX            ; ANHAENGEN
1230:  C3FA A9 50            LDA  #"P"
1240:  C3FC 9D 41 03         STA  CASPUF+5,X
1250:  C3FF E8               INX
1260:  C400 A9 2C            LDA  #","
1270:  C402 9D 41 03         STA  CASPUF+5,X
1280:  C405 E8               INX
1290:  C406 A9 57            LDA  #"W"
1300:  C408 9D 41 03         STA  CASPUF+5,X
1310:  C40B E8               INX
1320:  C40C 8A               TXA            ; LAENGE DES FILENAMEN
1330:  C40D A2 41            LDX  #$41      ; STARTADR DES     ' '
1340:  C40F A0 03            LDY  #$03
1350:  C411 20 BD FF         JSR  SETNAM
1360:  C414 A9 08            LDA  #$08      ; FILENUMMER
1370:  C416 A2 08            LDX  #$08      ; GERAETENUMMER
1380:  C418 A0 01            LDY  #$01      ; SECUNDAERADR.
1390:  C41A 20 BA FF         JSR  SETLFS
1400:  C41D 20 C0 FF         JSR  OPEN
1410:  C420 A2 08            LDX  #$08
1420:  C422 20 C9 FF         JSR  CHKOUT
1430:  C425 A9 03            LDA  #<CSTART
1440:  C427 A2 08            LDX  #>CSTART
1450:  C429 85 AC            STA  STARTV
1460:  C42B 86 AD            STX  STARTV+1
1470:  C42D AD 3E 03         LDA  CASPUF+2  ; ENDADRESSE BERECHNEN
1480:  C430 38               SEC            ; AUS DIFFERENZ START- + ENDADR.
1490:  C431 ED 3C 03         SBC  CASPUF    ; AUS DEM HEADER PLUS BESTIMMTER
1500:  C434 08               PHP            ; STARTADERSSE
1510:  C435 18               CLC
1520:  C436 65 AC            ADC  STARTV
1530:  C438 85 AE            STA  ENDVEC
1540:  C43A AD 3F 03         LDA  CASPUF+3
1550:  C43D 65 AD            ADC  STARTV+1
1560:  C43F 28               PLP
1570:  C440 ED 3D 03         SBC  CASPUF+1
1580:  C443 85 AF            STA  ENDVEC+1
```

```
1590:  C445 AD 3C 03          LDA  CASPUF   ; PROGRAMMSTARTADR.
1600:  C448 20 D2 FF          JSR  BSOUT    ; ZUR DISK SENDEN
1610:  C44B AD 3D 03          LDA  CASPUF+1
1620:  C44E 20 D2 FF          JSR  BSOUT
1622:  C451
1622:  C451
1623:                  ; SPEICHERSCHLEIFE
1624:  C451
1624:  C451
1630:  C451 A0 00     PSAVE  LDY  #$00
1640:  C453 B1 AC            LDA  (STARTV),Y
1650:  C455 20 D2 FF         JSR  BSOUT
1660:  C458 E6 AC            INC  STARTV   ; ADRESSE ERHOEHEN
1670:  C45A D0 02            BNE  NOTHI
1680:  C45C E6 AD            INC  STARTV+1
1690:  C45E A5 AC     NOTHI  LDA  STARTV
1700:  C460 C5 AE            CMP  ENDVEC   ; ENDADRESSE ERREICHT
1710:  C462 A5 AD            LDA  STARTV+1
1720:  C464 E5 AF            SBC  ENDVEC+1
1730:  C466 90 E9            BCC  PSAVE
1740:  C468 20 CC FF         JSR  CLRCH
1750:  C46B A9 08            LDA  #$08
1760:  C46D 4C C3 FF         JMP  CLOSE
1762:  C470
1762:  C470
1764:  C470
1764:  C470
1766:                  ; VON CC LADEN
1768:  C470
1768:  C470
1770:  C470 A2 00     CLOAD  LDX  #$00
1780:  C472 A0 03            LDY  #<CSTART
1790:  C474 A9 08            LDA  #>CSTART
1800:  C476 86 0A            STX  $0A      ; LOAD/VERIFY FLAG
1810:  C478 86 93            STX  $93
1820:  C47A 84 AC            STY  $AC
1830:  C47C 85 AD            STA  $AD
1840:  C47E A9 05            LDA  #1+4     ; AN BESTIMMTE ADRESSE,
1850:  C480 85 02            STA  ABSFLG   ; NICHT WARTEN
1860:  C482 A9 00            LDA  #$00     ; KEINEN FILENAMEN
1870:  C484 20 BD FF         JSR  SETNAM
1880:  C487 A2 01            LDX  #$01     ; GA
1890:  C489 A0 00            LDY  #$00     ; SA
1900:  C48B 20 BA FF         JSR  SETLFS
1910:  C48E 20 8E C1         JSR  LOADR
1922:  C491
1922:  C491
1924:                  ; STATUS TESTEN
1926:  C491
1926:  C491
1930:  C491 20 B7 FF HOLSTA  JSR  STATUS
1940:  C494 29 BF            AND  #$BF
1950:  C496 F0 05            BEQ  OK
1960:  C498 A2 1D            LDX  #$1D
1970:  C49A 4C 37 A4         JMP  ERROR
1980:  C49D 60        OK     RTS
ÜC3C0-C49E
```

```
100 GOSUB 240
110 PRINT"[CLEAR]**********************[LEFT2,DOWN]*[UP
    ]*"
120 PRINT"[UP]*BACKUP MIT FASTTAPE **[LEFT2,DOWN]*";
130 PRINT"*   VON CC AUF DISK   **[LEFT2,DOWN]*";
140 PRINT"*(C) DIRK PAULISSEN  **[LEFT2,DOWN]*";
150 PRINT"*********************"
160 PRINT:PRINT:PRINT"WIEVIELE PROGRAMME WOLLEN SIE"
170 PRINT:INPUT"COPIEREN ";AZ
180 POKE 767,AZ
190 PRINT:PRINT:PRINT"LEGEN SIE DIE QUELLCASSETTE EIN"
200 PRINT"[DOWN]UND DRUECKEN SIE << RETURN >>"
210 GET A$:IF A$<>CHR$(13)THEN 210
220 SYS 29632
230 END
************************************************************
240 REM LADEPROGRAMM  FASTCOPY T-D 20
************************************************************
250 E=29853:A=29632:PS=0
260 FOR I=A TO E:READ X:POKE I,X:PS=PS+X:NEXT
280 IF PS<>27173 THEN PRINT"FEHLER IN DATAS":END
290 RETURN
300 DATA 169,0,32,66,198,32,112,116,32,212,115,206,255,2
    ,240,3,76,192,115,96
310 DATA 169,16,162,65,160,3,32,189,255,32,44,247,169,13,
    32,210,255,162,16,202
320 DATA 189,65,3,201,32,208,4,224,0,208,244,232,169,44,
    157,65,3,232,169,80
330 DATA 157,65,3,232,169,44,157,65,3,232,169,87,157,65,
    3,232,138,162,65,160
340 DATA 3,32,189,255,169,8,162,8,160,1,32,186,255,32,19
    2,255,162,8,32,201,255
350 DATA 169,3,162,18,133,172,134,173,173,62,3,56,237,60,
    3,8,24,101,172,133
360 DATA 174,173,63,3,101,173,40,237,61,3,133,175,173,60
    ,3,32,210,255,173,61
370 DATA 3,32,210,255,160,0,177,172,32,210,255,230,172,2
    08,2,230,173,165,172
```

```
380 DATA 197,174,165,173,229,175,144,233,32,204,255,169,
    8,76,195,255,162,0,160
390 DATA 3,169,18,134,10,134,147,132,172,133,173,169,5,1
    33,2,169,0,32,189,255
400 DATA 162,1,160,0,32,186,255,32,136,113,32,183,255,41,
    191,240,5,162,29,76
410 DATA 55,196,96
```

```
100 GOSUB 240
110 PRINT"[CLEAR]***************************************
    *[LEFT,DOWN]*[UP]*"
120 PRINT"* B A C K U P   M I T   F A S T T A P E **[LEFT
    2,DOWN]*";
130 PRINT"*         VON CASSETTE AUF DISKETTE      **[LEFT
    2,DOWN]*";
140 PRINT"*            (C) DIRK PAULISSEN          **[LEFT
    2,DOWN]*";
150 PRINT"**************************************"
160 PRINT:PRINT:PRINT"WIEVIELE PROGRAMME WOLLEN SIE"
170 PRINT:INPUT"COPIEREN ";AZ
180 POKE 767,AZ
190 PRINT:PRINT:PRINT"LEGEN SIE DIE QELLCASSETTE EIN"
200 PRINT"[DOWN]UND DRUECKEN SIE << RETURN >>"
210 GET A$:IF A$<>CHR$(13)THEN 210
220 SYS 50112
230 END
************************************************************
240 REM LADEPROGRAMM  FASTCOPY T-D 64
************************************************************
250 E=50333:A=50112:PS=0
260 FOR I=A TO E:READ X:POKE I,X:PS=PS+X:NEXT
270 IF PS<>27517 THEN PRINT"FEHLER IN DATAS":END
280 RETURN
290 DATA 169,0,32,66,166,32,112,196,32,212,195,206,255,2
    ,240,3,76,192,195,96
300 DATA 169,16,162,65,160,3,32,189,255,32,147,246,169,1
    3,32,210,255,162,16
310 DATA 202,189,65,3,201,32,208,4,224,0,208,244,232,169,
    44,157,65,3,232,169
320 DATA 80,157,65,3,232,169,44,157,65,3,232,169,87,157,
    65,3,232,138,162,65
330 DATA 160,3,32,189,255,169,8,162,8,160,1,32,186,255,3
    2,192,255,162,8,32,201
340 DATA 255,169,3,162,8,133,172,134,173,173,62,3,56,237,
    60,3,8,24,101,172,133
350 DATA 174,173,63,3,101,173,40,237,61,3,133,175,173,60
```

```
    ,3,32,210,255,173,61
360 DATA 3,32,210,255,160,0,177,172,32,210,255,230,172,2
    08,2,230,173,165,172
370 DATA 197,174,165,173,229,175,144,233,32,204,255,169,
    8,76,195,255,162,0,160
380 DATA 3,169,8,134,10,134,147,132,172,133,173,169,5,13
    3,2,169,0,32,189,255
390 DATA 162,1,160,0,32,186,255,32,142,193,32,183,255,41,
    191,240,5,162,29,76
400 DATA 55,164,96
```

15.2 PROGRAMMBESCHREIBUNG BACKUP CC-DISK

Für diejenigen Leser, die zumindest Grundkenntnisse in ASSEMBLER-Programmierung haben, möchte ich das ASSEMBLER-Listing etwas näher erklären.

Die Hauptschleife des Programms steht ab $C3OCO bis $C3D3. Zuerst werden durch den NEW-Befehl alle Basic-Vektoren zurückgesetzt. Dann wird zur Cassettenloadroutine CLOAD verzweigt; nach der Ausführung dieses Unterprogramms wird das Programm durch DSAVE auf Disk gespeichert. Nachdem ANZAHL dekrementiert und getestet wurde, wird zum Start zurückgesprungen.

CLOAD ($C469)

Der Pufferstart CSTART wird in die Speicherstellen $AC, $AD übertragen. Aus diesen Speicherstellen holt sich FastTape die erforderliche Ladeadresse. Weiterhin werden die LOAD/VERIFY-Flags auf 0 = LOAD gesetzt. Durch das Setzen von Bit null und zwei in ABSFLG wird das FastTape-Programm veranlaßt, an die Adresse zu laden, die in $AC und $AD übergeben wird, und nach dem Finden des Programms nicht zu stoppen.

Ab $C47B werden die notwendigen Vorbereitungsroutinen des Betriebssystems SETNAM und SETLFS aufgerufen.

Danach wird zum FastTape-Unterprogramm LOADR verzweigt. Nach dem Ladevorgang wird der STATUS getestet und in die Hauptschleife zurückgesprungen.

DSAVE

Das Speichern auf Diskette ist etwas komplizierter. Zuerst wird über Betriebssystemroutinen "SAVING name" ausgegeben. Von $C3DE bis $C3EA wird das Ende des Filenamens, der im Cassettenpuffer steht, festgestellt, indem vom Ende her

solange auf "SPACE" getestet wird, bis ein anderes Zeichen gefunden wird.

In den folgenden Zeilen bis $C401 wird ",P,W" (Programm, Write) an den Filenamen angehängt. Über das X-Register wird die Länge des kompletten Filenamen bestimmt.

Ab $C465 beginnen die notwendigen Vorbereitungsroutinen zur Diskettenspeicherung SETNAM, SETLFS, OPEN, CHKOUT. Danach wird der Startvektor wieder auf den Pufferstart gesetzt. Die Endadresse des eingeladenen Programms wird aus der Differenz von Start und Endadresse aus dem Cassettenpuffer plus Startadresse des Puffers berechnet ($C426 - $C43C). Bei der Diskettenspeicherung geben die ersten beiden Bytes die Adresse an, von der es abgespeichert wurde. Diese Adresse steht als Startadresse im Cassettenpuffer. Sie wird in $CH3E - $C447 an die Diskette übermittelt.

Ab PSAVE ($C44A) beginnt die eigentliche Programmspeicherung. In das Y-Register wird 0 geladen, und die Programmbytes werden indirekt-indiziert in den Akkumulator geladen und über BASOUT an die Diskettenstation gesandt. Danach wird der Startvektor inkrementiert und mit dem Endvektor verglichen. Sind sie nicht gleich, wird wieder nach PSAVE gesprungen.

Sind die Vektoren gleich, wird der Diskettenausgabekanal geschlossen und zur Hauptschleife zurückgesprungen.

15.3 BACKUP VON DISKETTE AUF CASSETTE

Mit diesem Programm haben Sie die Möglichkeit, sehr komfortabel ein oder mehrere Programme von einer Diskette auf eine oder mehrere Cassetten zu übertragen. Die Eingabe erfolgt über einen Basiclader, den Sie am Anschluß an die Programmbeschreibung finden. Vergessen Sie auch bei diesem Basiclader nicht, ihn vor dem Starten abzuspeichern.

Dieses Programm läuft ebenfalls nur, wenn Sie FastTape schon geladen haben.

BEDIENUNG DES PROGRAMMS

Gestartet wird das Programm durch SYS 50176 (SYS 29696 beim VC 20). Sofort erscheint der Titel auf dem Bildschirm, und Sie werden gebeten, die Quelldiskette einzulegen und eine Taste durchzudrücken.

Nach kurzer Zeit erscheint der Diskettentitel und der erste Fileeintrag des Disketteninhaltsverzeichnisses mit der Frage JA/NEIN. Wenn Sie das Programm kopieren wollen, drücken Sie "J", und es erscheint JA hinter dem Fileeintrag. Dann wird der nächste Eintrag angezeigt. Wollen Sie ein Programm nicht kopieren, drücken Sie "N", und der Eintrag wird mit NEIN gekennzeichnet. Auf diese Weise können Sie bis zu 48 Programme mit JA kennzeichnen.

Falls Sie sich innerhalb des Programms einmal vertippen, können Sie durch Drücken der RUN/STOP-Taste immer das Programm abbrechen und zur Titelseite zurückkehren.

Wenn Sie das ganze Inhaltsverzeichnis bearbeitet haben, werden Sie gefragt, ob Sie die Programme einzeln oder durchgehend kopieren wollen. Möchten Sie alle Programme auf einer Diskette speichern, geben sie "1" ein. Wenn Sie eine "2" eingeben, sagt Ihnen der Computer immer, welches Programm er als nächstes einliest und wartet auf einen Tastendruck. Damit ist es möglich, für jedes Programm eine andere Cassette einzulegen.

Am Ende des Kopiervorgangs kehrt der Rechner wieder zur Titelseite zurück. Sie haben dann die Möglichkeit, eine neue Diskette zu kopieren oder durch Drücken von "E" ins Basic zurückzukehren.

```
60:     C400                    .OPT P1
                          ;*********************************
                          ;*                              *
                          ;*          BACKUP              *
                          ;*                              *
                          ;*  VON DISKETTE AUF CASSETTE   *
                          ;*                              *
                          ;*       MIT FASTTAPE           *
                          ;*                              *
                          ;*********************************
                          ;
                          ; START MIT SYS 50176 (SYS 29696)
134:    C400
                          ; WERTE UND KOMMENTARE IN KLAMMERN
136:    C400
                          ;        GELTEN FUER VC-20
                          ;
142:    C400                    *=    $C400   ; ($7400)
145:    C400
145:    C400             START   =    *
150:    C400             CLSCRN  =    $E544   ;($E55F) BILDSCHIRM LOESCHEN
152:    C400             NUMOUT  =    $BDCD   ;($DDCD) POS. INEGERZAHL AUSGEB.
154:    C400             MOTAUS  =    $FCCA   ;($FD08) MOTOR AUSSCHALTEN
156:    C400             PTASTE  =    $F838   ;($F8B7) RECORDERTASTE ABFRAGEN
158:    C400             SFINUM  =    $F30F   ;($F3CF) SUCHT FILENUMMER
160:    C400             STROUT  =    $AB1E   ;($CB1E) STRING AUSGEBEN
162:    C400             SETPAR  =    $F31F   ;($F3DF) SETZT FILEPARAMETER
164:    C400             ABSOLUT =    $C085   ;($707F) EINSPRUNG IN FASTTAPE
169:    C400
169:    C400
169:    C400
170:    C400             GETBYT  =    $FFE4   ;BYT VOM EINGABEGERAET HOHLEN
180:    C400             SETLFS  =    $FFBA   ;FILEPARAMETER SETZEN
190:    C400             SETNAM  =    $FFBD   ;FILENAMEPARAMETER SETZEN
200:    C400             OPEN    =    $FFC0
210:    C400             CLOSE   =    $FFC3
220:    C400             TALK    =    $FFB4
230:    C400             SETALK  =    $FF96   ;SECUNDAERADR. NACH TALK
240:    C400             IECIN   =    $FFA5   ;BYTE VOM IEC-BUS HOLEN
250:    C400             CLALL   =    $FFE7
260:    C400             UNTALK  =    $FFAB   ;UNTALK AUSGEBEN
280:    C400             STOP    =    $FFE1   ;STOPTASTE ABFRAGEN
290:    C400             BASOUT  =    $FFD2   ;BYT AUSGEBEN
300:    C400
300:    C400
300:    C400
320:    C400             GA      =    $BA     ;GERAETEADRESSE
330:    C400             SA      =    $B9     ;SEKUNDAERADRESSE
340:    C400             MAXBLK  =    $98     ;($5D) MAX LADBARE BLOECKE
350:    C400             CR      =    $0D     ;CARRADGE RETURN
360:    C400             COPANZ  =    $FB     ;AKTUELLE LISTENLAENGE
370:    C400             TEMP    =    $FC     ;HILFSPUFFER
380:    C400             FLAG    =    $FE     ;EINZEL FLAG
390:    C400             LISTPT  =    $41     ;POINTER AUF NAMENLISTE
400:    C400             FILPUF  =    $340    ;FILEPUFFER
410:    C400             LNGTAB  =    START+$640  ;TAB. DER NAMENLAENGEN
420:    C400             NAMTAB  =    START+$700  ;TAB. DER FILENAMEN
430:    C400             LSTMAX  =    $30     ;MAXIMALE LISTENLAENGE
440:    C400             TLSADR  =    $09     ;($13) HIBYTE DER LADEADRESSE
```

```
450:    C400             TEMP2      =     $22        ;HILFSPOINTER
460:    C400             TPGSTA     =     $AC        ;PTR ZUM TATS. PRGSTART
470:    C400             TPGEND     =     $AE        ;PTR ZUM TATS. PRGENDE
480:    C400             PRGSTA     =     $A7        ;PTR ZUR EIGENTL. PRGSTART
490:    C400             PRGEND     =     $A9        ;PTR ZUM EIGENTL. PRGENDE
495:    C400             MOFLAG     =     $C0        ;MOTORFLAG
                          ;
                          ;
560:    C400 A9 01                  LDA   #1        ;(LDA #25)
570:    C402 8D 20 D0               STA   $D020     ;(STA $900F)
580:    C405 8D 21 D0               STA   $D021     ;(ENTFAELLT)
590:    C408 A9 06                  LDA   #6
600:    C40A 8D 86 02               STA   $286
610:    C40D A9 6B                  LDA   #<TITEL
620:    C40F A0 C8                  LDY   #>TITEL
630:    C411 20 1E AB               JSR   STROUT
640:    C414 A9 34                  LDA   #<QUEINS  ;"QUELLDISK EINSTECKEN "
650:    C416 A0 C8                  LDY   #>QUEINS
660:    C418 20 1E AB               JSR   STROUT
670:    C41B 20 99 C7               JSR   WAIT
680:    C41E C9 45                  CMP   #"E"      ;PROGRAMM BEENDEN
690:    C420 D0 01                  BNE   OKCOPY
700:    C422 60                     RTS
701:    C423
701:    C423
                          ; INHALT EINLESEN
703:    C423
703:    C423
710:    C423 20 B6 C7 OKCOPY        JSR   CARET
720:    C426 20 B6 C7               JSR   CARET
730:    C429 20 9F C7               JSR   INIT      ;DISK INITIALISIEREN
740:    C42C A9 08                  LDA   #$08
750:    C42E AA                     TAX
760:    C42F A0 00                  LDY   #$00
770:    C431 20 BA FF               JSR   SETLFS
780:    C434 A9 01                  LDA   #$01
790:    C436 A2 E1                  LDX   #<CATALO
800:    C438 A0 C9                  LDY   #>CATALO
810:    C43A 20 BD FF               JSR   SETNAM
820:    C43D 20 C0 FF               JSR   OPEN
830:    C440 A9 08                  LDA   #$08
840:    C442 20 B4 FF               JSR   TALK
850:    C445 A9 00                  LDA   #$00
860:    C447 20 96 FF               JSR   SETALK
870:    C44A A0 04                  LDY   #$04
880:    C44C 20 A5 FF LOOP1         JSR   IECIN     ;DIE ERSTEN 4 BYTES EINLESEN
890:    C44F 88                     DEY
900:    C450 D0 FA                  BNE   LOOP1
910:    C452 20 A5 FF               JSR   IECIN     ;BLOCKZAHL EINLESEN
920:    C455 85 22                  STA   TEMP2
930:    C457 20 A5 FF               JSR   IECIN
940:    C45A A6 22                  LDX   TEMP2
950:    C45C 20 CD BD               JSR   NUMOUT    ;UND AUSGEBEN
960:    C45F 20 B3 C7               JSR   SPACE
970:    C462 20 A5 FF LOOP2         JSR   IECIN     ;DISKNAME ETC AUSGEBEN
980:    C465 F0 06                  BEQ   NULL
990:    C467 20 D2 FF               JSR   BASOUT
1000:   C46A 18                     CLC
1010:   C46B 90 F5                  BCC   LOOP2
1020:   C46D 20 B6 C7 NULL          JSR   CARET
1030:   C470 20 B6 C7               JSR   CARET
```

```
1040:  C473 20 A5 FF          JSR  IECIN
1050:  C476 20 A5 FF          JSR  IECIN
1060:  C479 A0 00             LDY  #$00     ;ANZAHL DER ZU COPIERENDEN
1070:  C47B 84 FB             STY  COPANZ   ;FILES AUF NULL SETZEN
1071:  C47D
1071:  C47D
                    ; INHALT AUSGEBEN UND FRAGEN
1073:  C47D
1073:  C47D
1080:  C47D 20 A5 FF LSTART   JSR  IECIN
1090:  C480 85 FC             STA  TEMP
1100:  C482 20 A5 FF          JSR  IECIN
1110:  C485 85 FD             STA  TEMP+1
1120:  C487 A6 FC             LDX  TEMP
1130:  C489 20 CD BD          JSR  NUMOUT   ;BLOCKZAHL AUSGEBENN
1140:  C48C 20 B3 C7          JSR  SPACE
1150:  C48F A0 00             LDY  #$00     ;FILENAMEN
1160:  C491 20 A5 FF WRFILE   JSR  IECIN    ;EINLESEN
1170:  C494 48                PHA
1180:  C495 20 BB C7          JSR  AUSGAB   ;AUSGEBEN
1190:  C498 68                PLA           ;UND IN
1200:  C499 99 40 03          STA  FILPUF,Y  ;PUFFER SCHREIBEN
1210:  C49C F0 03             BEQ  NAMEND
1220:  C49E C8                INY
1230:  C49F D0 F0             BNE  WRFILE
1240:  C4A1 20 A5 FF NAMEND   JSR  IECIN
1250:  C4A4 20 A5 FF          JSR  IECIN
1260:  C4A7 A5 90             LDA  $90      ;STATUS ABFRAGEN
1270:  C4A9 F0 03             BEQ  FRAGEN
1280:  C4AB 4C 47 C5          JMP  INHEND
1290:  C4AE A5 FD    FRAGEN   LDA  TEMP+1
1300:  C4B0 D0 06             BNE  TOLONG
1310:  C4B2 A5 FC             LDA  TEMP
1320:  C4B4 C9 98             CMP  #MAXBLK   ;AUF MAXIMALE LAENGE
1330:  C4B6 90 0A             BCC  LNGEOK   ;TESTEN
1340:  C4B8 A9 FC    TOLONG   LDA  #<SLONG   ;"ZU LANG ZUM
1350:  C4BA A0 C8             LDY  #>SLONG   ;ZUM COPIEREN"
1360:  C4BC 20 1E AB          JSR  STROUT   ;AUSGEBEN
1370:  C4BF 4C 41 C5 FINISH   JMP  NXFILE
1380:  C4C2 A5 FB    LNGEOK   LDA  COPANZ
1390:  C4C4 C9 30             CMP  #LSTMAX
1400:  C4C6 90 0A             BCC  FANZOK
1410:  C4C8 A9 1B             LDA  #<LILONG  ;"LISTE ZU LANGE"
1420:  C4CA A0 C9             LDY  #>LILONG
1430:  C4CC 20 1E AB          JSR  STROUT   ;AUSGEBEN
1440:  C4CF 18                CLC
1450:  C4D0 90 ED             BCC  FINISH
1460:  C4D2 A9 00    FANZOK   LDA  #$00     ;"JA/NEIN" AUSGEBEN
1470:  C4D4 85 08             STA  $08
1480:  C4D6 A9 1F             LDA  #$1F     ;(ENTFAELLT) TABULIERUNG
1490:  C4D8 85 D3             STA  $D3      ;(ENTFAELLT) UEBER SPALTENZEIGER
1500:  C4DA A9 33             LDA  #<JANEIN
1510:  C4DC A0 C9             LDY  #>JANEIN
1520:  C4DE 20 1E AB          JSR  STROUT
1530:  C4E1 20 E1 C7 LOOP3    JSR  INPUT
1540:  C4E4 C9 4E             CMP  #"N"
1550:  C4E6 F0 52             BEQ  NEIN
1560:  C4E8 C9 4A             CMP  #"J"
1570:  C4EA D0 F5             BNE  LOOP3
1580:  C4EC A9 45             LDA  #<STRJA   ;JA AUSGEBEN
1590:  C4EE A0 C9             LDY  #>STRJA
```

```
1600:   C4F0 20 1E AB           JSR  STROUT
1610:   C4F3 A5 FB              LDA  COPANZ
1620:   C4F5 20 F1 C6           JSR  SETPTR
1630:   C4F8 A2 00              LDX  #$00
1640:   C4FA E8        LOOP4    INX             ;FILENAMENSTART SUCHEN
1650:   C4FB BD 40 03           LDA  FILPUF,X
1660:   C4FE C9 22              CMP  #$22       ;1.HOCHKOMMA = START
1670:   C500 D0 F8              BNE  LOOP4
1680:   C502 86 FD              STX  TEMP+1     ;START RETTEN
1690:   C504 E8                 INX
1700:   C505 BD 40 03 UMSETZ    LDA  FILPUF,X   ;NAMEN IN LISTE SCHREIBEN
1710:   C508 C9 22              CMP  #$22       ;2. HOCHKOMMA = ENDE
1720:   C50A F0 06              BEQ  UMSEND
1730:   C50C 91 41              STA  (LISTPT),Y
1740:   C50E E8                 INX
1750:   C50F C8                 INY
1760:   C510 D0 F3              BNE  UMSETZ
1770:   C512 8A        UMSEND   TXA             ;ENDPUNKT IN AKKU
1780:   C513 A4 FB              LDY  COPANZ     ;INDEX IN Y-REG.
1790:   C515 18                 CLC
1800:   C516 E5 FD              SBC  TEMP+1     ;LAENGA BERECHNEN
1810:   C518 99 40 CA           STA  LNGTAB,Y   ;IN TABELLE SCHREIBEN
1820:   C51B BD 40 03 WETEST    LDA  FILPUF,X
1830:   C51E D0 0A              BNE  TYPTST
1840:   C520 A9 51     ILL      LDA  #<STRILL   ;"ILLEGALER FILETYP"
1850:   C522 A0 C9              LDY  #>STRILL
1860:   C524 20 1E AB           JSR  STROUT     ;AUSGEBEN
1870:   C527 4C 41 C5           JMP  NXFILE
1880:   C52A C9 53     TYPTST   CMP  #"S"       ;TESTET AUF PRG FILE
1890:   C52C F0 F2              BEQ  ILL
1900:   C52E C9 50              CMP  #"P"
1910:   C530 F0 03              BEQ  TYPOK
1920:   C532 E8                 INX
1930:   C533 D0 E6              BNE  WETEST
1940:   C535 E6 FB     TYPOK    INC  COPANZ     ;ANZ DER ZU COPIERENDEN
1950:   C537 18                 CLC             ;FILES ERHOEHEN
1960:   C538 90 07              BCC  NXFILE
1970:   C53A A9 6B     NEIN     LDA  #<STRNO    ;"NEIN" AUSGEBEN
1980:   C53C A0 C9              LDY  #>STRNO
1990:   C53E 20 1E AB           JSR  STROUT
2000:   C541 20 B6 C7 NXFILE    JSR  CARET
2010:   C544 4C 7D C4           JMP  LSTART
2020:   C547 A9 08     INHEND   LDA  #$08       ;FILE SCHLIESSEN
2030:   C549 20 C3 FF           JSR  CLOSE
2040:   C54C A5 FB              LDA  COPANZ     ;SOLLEN FILES COPIERT
2050:   C54E D0 03              BNE  COPYST     ;WERDEN/NEIN=> START
2060:   C550 4C 00 C4           JMP  START
2062:   C553
2062:   C553
2062:   C553
                        ;COPIEREN VORBEREITEN
2066:   C553
2066:   C553
2066:   C553
2070:   C553 A9 77     COPYST   LDA  #<FRAGE    ;"EINZELN ODER
2080:   C555 A0 C9              LDY  #>FRAGE    ; HINTEREINANDER"
2090:   C557 20 1E AB           JSR  STROUT     ;AUSGEBEN
2100:   C55A 20 E1 C7 ABFRA2    JSR  INPUT
2110:   C55D C9 31              CMP  #"1"
2120:   C55F F0 07              BEQ  EINZEL
2130:   C561 C9 32              CMP  #"2"
```

```
2140:   C563 D0 F5                BNE   ABFRA2
2150:   C565 A9 00                LDA   #$00     ;FLAG ENTSPRECHEND
2160:   C567 2C                   .BYT  $2C
2170:   C568 A9 FF      EINZEL    LDA   #$FF     ;N
2180:   C56A 85 FE                STA   FLAG
2190:   C56C A9 0F                LDA   #$0F     ;FEHLERKANAL SCHLIESSEN
2200:   C56E 20 C3 FF             JSR   CLOSE
2202:   C571
2202:   C571
2202:   C571
                        ;COPIER - ROUTINE
2206:   C571
2206:   C571
2206:   C571
                        ;EINLESEN VON DISK
                        ;   VORBEREITEN
2210:   C571
2210:   C571 A2 00                LDX   #$00     ;TEMP LOESCHEN
2220:   C573 86 FC                STX   TEMP     ;TEMP => PRG-ZAELER
2230:   C575 A9 C7      LOLOOP    LDA   #<SREAD  ;"READING"
2240:   C577 A0 C9                LDY   #>SREAD
2250:   C579 20 1E AB             JSR   STROUT   ; AUSGEBEN
2260:   C57C A4 FC                LDY   TEMP     ;POS. IN TABELLE
2270:   C57E BE 40 CA             LDX   LNGTAB,Y  ;LAENGE NACH X-REG.
2280:   C581 A5 FC                LDA   TEMP
2290:   C583 20 F1 C6             JSR   SETPTR
2300:   C586 B1 41      WRTNAM    LDA   (LISTPT),Y  ;FILENAME AUSGEBEN
2310:   C588 20 D2 FF             JSR   BASOUT
2320:   C58B C8                   INY
2330:   C58C CA                   DEX
2340:   C58D D0 F7                BNE   WRTNAM
2350:   C58F A9 02                LDA   #$02     ;FILENUMMER
2360:   C591 A2 08                LDX   #$08     ;GA
2370:   C593 A8                   TAY            ;SA
2380:   C594 20 BA FF             JSR   SETLFS
2390:   C597 A6 FC                LDX   TEMP
2400:   C599 BD 40 CA             LDA   LNGTAB,X
2410:   C59C 85 22                STA   TEMP2
2420:   C59E A5 FC                LDA   TEMP
2430:   C5A0 20 F1 C6             JSR   SETPTR
2440:   C5A3 A2 00                LDX   #$00
2450:   C5A5 B1 41      LOOP6     LDA   (LISTPT),Y  ;FILENAME IN PUFFER
2460:   C5A7 9D 40 03             STA   FILPUF,X  ;SCHREIBEN
2470:   C5AA C8                   INY
2480:   C5AB E8                   INX
2490:   C5AC C6 22                DEC   TEMP2
2500:   C5AE D0 F5                BNE   LOOP6
2510:   C5B0 A0 00                LDY   #$00
2520:   C5B2 B9 D2 C9   LOOP7     LDA   PRGRE,Y  ; ",P,R " ANHAENGEN
2530:   C5B5 9D 40 03             STA   FILPUF,X
2540:   C5B8 C8                   INY
2550:   C5B9 E8                   INX
2560:   C5BA C0 04                CPY   #$04
2570:   C5BC 90 F4                BCC   LOOP7
2580:   C5BE 8A                   TXA
2590:   C5BF A2 40                LDX   #<FILPUF
2600:   C5C1 A0 03                LDY   #>FILPUF
2610:   C5C3 20 BD FF             JSR   SETNAM
2620:   C5C6 20 C0 FF             JSR   OPEN
2630:   C5C9 A9 09                LDA   #TLSADR
2640:   C5CB A0 00                LDY   #$00
```

```
2650:   C5CD 84 22              STY  TEMP2
2660:   C5CF 85 23              STA  TEMP2+1
2670:   C5D1 AD 11 D0           LDA  $D011   ;(ZEILE 2670-2690 ENTFALLEN)
2680:   C5D4 29 EF              AND  #$EF    ;BILDSCHIRM AUS
2690:   C5D6 8D 11 D0           STA  $D011
2700:   C5D9 A9 0F              LDA  #$0F    ;KANAL 15 OEFFNEN
2710:   C5DB A2 08              LDX  #$08
2720:   C5DD A8                 TAY
2730:   C5DE 20 BA FF           JSR  SETLFS
2740:   C5E1 A9 03              LDA  #$03    ;"UI-" DISKETTE
2750:   C5E3 A2 D7              LDX  #<UIMIN  ;SCHNELL SCHALTEN
2760:   C5E5 A0 C9              LDY  #>UIMIN
2770:   C5E7 20 BD FF           JSR  SETNAM
2780:   C5EA 20 C0 FF           JSR  OPEN
2790:   C5ED A2 02              LDX  #$02    ;LADEKANAL OEFFNEN
2800:   C5EF 20 0F F3           JSR  SFINUM
2810:   C5F2 20 1F F3           JSR  SETPAR
2820:   C5F5 A5 BA              LDA  GA
2830:   C5F7 20 B4 FF           JSR  TALK
2840:   C5FA A5 B9              LDA  SA
2850:   C5FC 20 96 FF           JSR  SETALK
2860:   C5FF A0 00              LDY  #$00
2862:   C601
2862:   C601
                    ;PROGRAMM IN PUFFER LADEN
2866:   C601
2866:   C601
2870:   C601 20 A5 FF LDLOOP    JSR  IECIN   ;PRG LADEN
2880:   C604 20 09 C7           JSR  STOBYT
2890:   C607 A6 90              LDX  $90
2900:   C609 F0 F6              BEQ  LDLOOP
2910:   C60B 20 EB C7           JSR  FEHLER
2920:   C60E 08                 PHP
2930:   C60F A9 02              LDA  #$02
2940:   C611 20 C3 FF           JSR  CLOSE
2950:   C614 AD 11 D0           LDA  $D011   ;(ENTFAELLT) BILDSCHIRM AN
2960:   C617 09 10              ORA  #$10    ;(ENTFAELLT)
2970:   C619 8D 11 D0           STA  $D011   ;(ENTFAELLT)
2980:   C61C 28                 PLP
2990:   C61D 90 03              BCC  NOFEHL
3000:   C61F 20 37 C7           JSR  FEMELD
3002:   C622
3002:   C622
                    ;AUF CC SPEICHERN VORBEREITEN
3006:   C622
3006:   C622
3010:   C622 A6 FC    NOFEHL    LDX  TEMP
3020:   C624 A5 22              LDA  TEMP2
3030:   C626 85 AE              STA  TPGEND
3040:   C628 A5 23              LDA  TEMP2+1
3050:   C62A 85 AF              STA  TPGEND+1
3060:   C62C E8                 INX
3070:   C62D A5 FC              LDA  TEMP
3075:   C62F D0 0D              BNE  NOMELD
3080:   C631 A9 4E              LDA  #<ZICASS   ;"CASSETTE EINLEGEN "
3090:   C633 A0 C8              LDY  #>ZICASS
3100:   C635 20 1E AB           JSR  STROUT
3105:   C638 20 E1 C7           JSR  INPUT
3110:   C63B 20 25 C8           JSR  TASTE   ;RECORDERTASTE ABFRAGEN
3115:   C63E A9 E3    NOMELD    LDA  #<WRITI
3120:   C640 A0 C9              LDY  #>WRITI
```

```
3125:   C642 20 1E AB             JSR   STROUT
3130:   C645 A4 FC                LDY   TEMP
3135:   C647 BE 40 CA             LDX   LNGTAB,Y
3140:   C64A A5 FC                LDA   TEMP
3145:   C64C 20 F1 C6             JSR   SETPTR
3150:   C64F B1 41       WRNAM2   LDA   (LISTPT),Y
3155:   C651 20 D2 FF             JSR   BASOUT
3160:   C654 C8                   INY
3160:   C655 CA                   DEX
3165:   C656 D0 F7                BNE   WRNAM2
3210:   C658 A2 01                LDX   #$01      ;FILEPARAMETER
3220:   C65A A9 00                LDA   #0        ;FUER CASSETTENSPEI-
3230:   C65C A8                   TAY             ;CHERUNG
3240:   C65D 20 BA FF             JSR   SETLFS
3250:   C660 A6 FC                LDX   TEMP
3260:   C662 BD 40 CA             LDA   LNGTAB,X
3270:   C665 85 22                STA   TEMP2
3280:   C667 A5 FC                LDA   TEMP
3290:   C669 20 F1 C6             JSR   SETPTR
3300:   C66C A2 00                LDX   #$00
3310:   C66E B1 41       ULOOP    LDA   (LISTPT),Y  ;FILENAMEN IN
3320:   C670 9D 40 03             STA   FILPUF,X  ;CASNPUFFER
3330:   C673 E8                   INX             ;SCHREIBEN
3340:   C674 C8                   INY
3350:   C675 C6 22                DEC   TEMP2
3360:   C677 D0 F5                BNE   ULOOP
3370:   C679 8A                   TXA             ;FILENAMEPARAMETER
3380:   C67A A2 40                LDX   #<FILPUF  ;N
3390:   C67C A0 03                LDY   #>FILPUF
3400:   C67E 20 BD FF             JSR   SETNAM
3410:   C681 A0 00                LDY   #$00
3420:   C683 A9 09                LDA   #TLSADR
3425:   C685 84 AC                STY   TPGSTA
3430:   C687 85 AD                STA   TPGSTA+1
3440:   C689 B1 AC                LDA   (TPGSTA),Y  ;PRGSTARTADR.
3450:   C68B 85 A7                STA   PRGSTA  ;IN PRGSTARTVEKTOR
3460:   C68D C8                   INY
3470:   C68E B1 AC                LDA   (TPGSTA),Y
3480:   C690 85 A8                STA   PRGSTA+1
3490:   C692 C8                   INY
3500:   C693 84 AC                STY   TPGSTA
3520:   C695 A5 AE                LDA   TPGEND  ;ENDADRESSE BERECHNEN
3530:   C697 38                   SEC
3540:   C698 E5 AC                SBC   TPGSTA
3550:   C69A 08                   PHP
3555:   C69B 18                   CLC
3560:   C69C 65 A7                ADC   PRGSTA
3570:   C69E 85 A9                STA   PRGEND
3580:   C6A0 A5 AF                LDA   TPGEND+1
3590:   C6A2 65 A8                ADC   PRGSTA+1
3600:   C6A4 28                   PLP
3610:   C6A5 E5 AD                SBC   TPGSTA+1
3620:   C6A7 85 AA                STA   PRGEND+1
3622:   C6A9
3622:   C6A9
                          ;PRG AUF CC SPEICHERN
3626:   C6A9
3626:   C6A9
3630:   C6A9 A2 05                LDX   #5        ;SYNCHRONISATIONS
3640:   C6AB 86 AB                STX   $AB       ;WIEDERHOLUNG
3650:   C6AD 20 85 C0             JSR   ABSOLUT   ;FASTTAPE SPEICHERN
```

```
3690:  C6B0 20 E5 C6          JSR  SCRON    ;(ENTFAELLT)
3700:  C6B3 E6 FC             INC  TEMP
3710:  C6B5 A6 FC             LDX  TEMP
3720:  C6B7 E4 FB             CPX  COPANZ
3730:  C6B9 B0 07             BCS  CENDEN
3732:  C6BB 24 FE             BIT  FLAG
3733:  C6BD 10 10             BPL  EINZE2
3740:  C6BF 4C DF C6          JMP  NXTFIL
3755:  C6C2 A9 ED    CENDEN   LDA  #<EOCOP   ;"ENDE DES COPIERENS"
3760:  C6C4 A0 C9             LDY  #>EOCOP
3770:  C6C6 20 1E AB          JSR  STROUT   ;AUSGEBEN
3780:  C6C9 20 E1 C7          JSR  INPUT
3790:  C6CC 4C 00 C4          JMP  START
3810:  C6CF 20 55 C7 EINZE2   JSR  NNAMAU
3820:  C6D2 A9 4E             LDA  #<ZICASS   ;"CASSETTE EINLEGEN "
3830:  C6D4 A0 C8             LDY  #>ZICASS
3840:  C6D6 20 1E AB          JSR  STROUT   ;AUSGEBEN
3845:  C6D9 20 E1 C7          JSR  INPUT    ;AUF TASTE WARTEN
3850:  C6DC 20 25 C8          JSR  TASTE
3870:  C6DF 20 B6 C7 NXTFIL   JSR  CARET
3880:  C6E2 4C 75 C5          JMP  LOLOOP   ;NAECHSTES FILE LADEN
3882:  C6E5
3882:  C6E5
3882:  C6E5
                     ;      UNTERPROGRAMME
3886:  C6E5
3886:  C6E5
3886:  C6E5
                     ;BILDSCHIRM ANSCHALTEN
3888:  C6E5
3888:  C6E5
3890:  C6E5 20 73 C7 SCRON    JSR  SEUIP    ;(ZEILE 3890-3930 ENTFALLEN)
3900:  C6E8 AD 11 D0          LDA  $D011
3910:  C6EB 09 10             ORA  #$10
3920:  C6ED 8D 11 D0          STA  $D011
3930:  C6F0 60                RTS
3932:  C6F1
3932:  C6F1
                     ;LISTENTHENER (LISTPT) AUF
                     ;AUF ENTSPR. FILENAMEN N
3936:  C6F1
3936:  C6F1
3940:  C6F1 A0 00    SETPTR   LDY  #$00
3950:  C6F3 0A                ASL  A
3960:  C6F4 0A                ASL  A
3970:  C6F5 84 42             STY  LISTPT+1
3980:  C6F7 0A                ASL  A
3990:  C6F8 26 42             ROL  LISTPT+1
4000:  C6FA 0A                ASL  A
4010:  C6FB 26 42             ROL  LISTPT+1
4020:  C6FD 85 41             STA  LISTPT
4030:  C6FF A5 42             LDA  LISTPT+1
4040:  C701 18                CLC
4050:  C702 69 CB             ADC  #>NAMTAB
4060:  C704 85 42             STA  LISTPT+1
4070:  C706 A0 00             LDY  #$00
4080:  C708 60                RTS
4082:  C709
4082:  C709
                     ;BYTE IN DEN SPEICHER SCHREIBEN
4086:  C709
```

```
4086:  C709
4090:  C709 91 22     STOBYT   STA  (TEMP2),Y
4100:  C70B E6 22              INC  TEMP2    ;ZAEHLER ERHOEHEN
4110:  C70D D0 02              BNE  NOINC
4120:  C70F E6 23              INC  TEMP2+1
4130:  C711 58        NOINC    CLI
4140:  C712 60                 RTS
4142:  C713
4142:  C713
                       ;ASCII-BYTE HOLEN UND IN HEXZAHL
                       ;         UMWANDELN
4146:  C713
4146:  C713
4150:  C713 20 A5 FF ASCHEX   JSR  IECIN
4160:  C716 29 0F              AND  #$0F
4170:  C718 0A                 ASL  A
4180:  C719 0A                 ASL  A
4190:  C71A 0A                 ASL  A
4200:  C71B 0A                 ASL  A
4210:  C71C 85 57              STA  $57
4220:  C71E 20 A5 FF           JSR  IECIN
4230:  C721 29 0F              AND  #$0F
4240:  C723 05 57              ORA  $57
4250:  C725 60                 RTS
4252:  C726
4252:  C726
                       ;HEXZAHL IN ASCII-BYTE UMWANDELN
                       ;        UND AUSGEBEN
4256:  C726
4256:  C726
4260:  C726 48        HEXASC   PHA
4270:  C727 4A                 LSR  A
4280:  C728 4A                 LSR  A
4290:  C729 4A                 LSR  A
4300:  C72A 4A                 LSR  A
4310:  C72B 20 2F C7           JSR  HALBBT
4320:  C72E 68                 PLA
4330:  C72F 29 0F     HALBBT   AND  #$0F
4340:  C731 18                 CLC
4350:  C732 69 30              ADC  #$30
4360:  C734 4C D2 FF           JMP  BASOUT   ;AUSGEBEN
4362:  C737
4362:  C737
                       ;FEHLERMELDUNG AUSGEBEN
4366:  C737
4366:  C737
4370:  C737 AD 11 D0 FEMELD    LDA  $D011    ;(ZEILE 4370-4390 ENTFALLEN)
4380:  C73A 09 10              ORA  #$10     ;(FMELD KOMMT IN ZEILE 4400)
4390:  C73C 8D 11 D0           STA  $D011
4400:  C73F A9 07              LDA  #<WMACH  ;"WEITER MACHEN ? "
4410:  C741 A0 CA              LDY  #>WMACH
4420:  C743 20 1E AB           JSR  STROUT   ;AUSGEBEN
4430:  C746 20 E1 C7 FLOOP     JSR  INPUT    ;AUF TASTE WARTEN
4440:  C749 C9 59              CMP  #$59
4450:  C74B D0 01              BNE  NEIN2
4460:  C74D 60                 RTS
4470:  C74E C9 4E     NEIN2    CMP  #$4E
4480:  C750 D0 F4              BNE  FLOOP
4490:  C752 4C D0 C7           JMP  ENDE
4492:  C755
4492:  C755
```

```
                          ;FOLGENDES FILE AUSGEBEN
4496:   C755
4496:   C755
4500:   C755 A9 1A    NNAMAU   LDA  #<SNFILE  ;"NAECHSTES FILE "
4510:   C757 A0 CA             LDY  #>SNFILE
4520:   C759 20 1E AB          JSR  STROUT   ;AUSGEBEN
4530:   C75C A5 FC             LDA  TEMP     ;POS IN TABELLE
4540:   C75E 0A                ASL  A        ;BERECHNEN
4550:   C75F 0A                ASL  A
4560:   C760 0A                ASL  A
4570:   C761 0A                ASL  A
4580:   C762 A6 FC             LDX  TEMP
4590:   C764 BC 40 CA          LDY  LNGTAB,X
4600:   C767 AA                TAX
4610:   C768 BD 00 CB WRNAM3   LDA  NAMTAB,X  ;FILENAMEN AUSGEBEN
4620:   C76B 20 D2 FF          JSR  BASOUT
4630:   C76E E8                INX
4640:   C76F 88                DEY
4650:   C770 D0 F6             BNE  WRNAM3
4660:   C772 60                RTS
4662:   C773
4662:   C773
                          ;DISKETTE AUF 'LANGSAM' SCHALTEN
                          ;  UND BILDSCHIRM ANSCHALTEN
4666:   C773
4666:   C773
4670:   C773 AD 11 D0 SEUIP    LDA  $D011    ;(ZEILE 4670-4820 ENTFALLEN)
4680:   C776 29 EF             AND  #$EF     ;BILDSCHIRM AN
4690:   C778 8D 11 D0          STA  $D011
4700:   C77B A9 0F             LDA  #$0F
4710:   C77D 20 C3 FF          JSR  CLOSE
4720:   C780 A9 0F             LDA  #$0F
4730:   C782 A2 08             LDX  #$08
4740:   C784 A8                TAY
4750:   C785 20 BA FF          JSR  SETLFS
4760:   C788 A9 03             LDA  #$03
4770:   C78A A2 DB             LDX  #<UIPLU
4780:   C78C A0 C9             LDY  #>UIPLU
4790:   C78E 20 BD FF          JSR  SETNAM
4800:   C791 20 C0 FF          JSR  OPEN
4810:   C794 A9 0F             LDA  #$0F
4820:   C796 4C C3 FF          JMP  CLOSE
4822:   C799
4822:   C799
                          ;  WARTESCHLEIFE
4826:   C799
4826:   C799
4830:   C799 20 E4 FF WAIT     JSR  GETBYT
4840:   C79C F0 FB             BEQ  WAIT
4850:   C79E 60                RTS
4852:   C79F
4852:   C79F
                          ;DISK INITIALISIEREN
4856:   C79F
4856:   C79F
4860:   C79F A9 0F    INIT     LDA  #$0F     ;KOMMANDOKANAL
4870:   C7A1 A2 08             LDX  #$08     ;OEFFNEN
4880:   C7A3 A8                TAY
4890:   C7A4 20 BA FF          JSR  SETLFS
4900:   C7A7 A9 01             LDA  #$01     ; "I"
4910:   C7A9 A2 DF             LDX  #<STRI
```

```
4920:  C7AB A0 C9              LDY  #>STRI
4930:  C7AD 20 BD FF           JSR  SETNAM
4940:  C7B0 4C C0 FF           JMP  OPEN
4942:  C7B3
4942:  C7B3
                         ;SPACE BZW. CARRIAGE RETURN SENDEN
4946:  C7B3
4946:  C7B3
4950:  C7B3 A9 20     SPACE    LDA  #$20     ;SPACE
4960:  C7B5 2C                 .BYT $2C
4970:  C7B6 A9 0D     CARET    LDA  #CR      ;CARRIDGE RETURN
4980:  C7B8 4C D2 FF           JMP  BASOUT   ;SENDEN
4982:  C7BB
4982:  C7BB
                         ;AUSGABEROUTINE
4986:  C7BB
4986:  C7BB
4990:  C7BB 20 D2 FF AUSGAB    JSR  BASOUT
5000:  C7BE 8A        STASTE   TXA           ;A,X,Y REGISTER RETTEN
5010:  C7BF 48                 PHA
5020:  C7C0 98                 TYA
5030:  C7C1 48                 PHA
5040:  C7C2 20 E1 FF           JSR  STOP     ;STOPTASTE ABFRAGEN
5050:  C7C5 18                 CLC
5060:  C7C6 D0 01              BNE  WEITER
5070:  C7C8 38                 SEC
5080:  C7C9 68        WEITER   PLA           ;REGISTER WIEDERHERSTELLEN
5090:  C7CA A8                 TAY
5100:  C7CB 68                 PLA
5110:  C7CC AA                 TAX
5120:  C7CD B0 01     STOP2    BCS  ENDE
5130:  C7CF 60        RETURN   RTS
5132:  C7D0
5132:  C7D0
                         ;       ABBRECHEN
5136:  C7D0
5136:  C7D0
5140:  C7D0 A2 F6     ENDE     LDX  #$F6
5150:  C7D2 9A                 TXS
5160:  C7D3 20 E5 C6           JSR  SCRON    ;(ENTFAELLT)
5170:  C7D6 A9 0F              LDA  #$0F
5180:  C7D8 20 C3 FF           JSR  CLOSE
5190:  C7DB 20 E7 FF           JSR  CLALL
5200:  C7DE 4C 00 C4           JMP  START
5202:  C7E1
5202:  C7E1
                         ;   EINGABEROUTINE
5206:  C7E1
5206:  C7E1
5210:  C7E1 20 E4 FF INPUT     JSR  GETBYT
5220:  C7E4 D0 E9              BNE  RETURN
5230:  C7E6 20 BE C7           JSR  STASTE
5240:  C7E9 90 F6              BCC  INPUT
5250:  C7EB A2 0F     FEHLER   LDX  #$0F     ;STATUS ABFRAGEN
5260:  C7ED 20 0F F3           JSR  SFINUM
5270:  C7F0 20 1F F3           JSR  SETPAR
5280:  C7F3 A5 BA              LDA  GA
5290:  C7F5 20 B4 FF           JSR  TALK
5300:  C7F8 A5 B9              LDA  SA
5310:  C7FA 20 96 FF           JSR  SETALK
5320:  C7FD 20 13 C7           JSR  ASCHEX
```

```
5330:   C800 C9 20              CMP  #$20
5340:   C802 08                 PHP
5350:   C803 90 0B              BCC  MELAUS   ;UND BEI FEHLER
5360:   C805 48                 PHA           ;AUSGEBEN
5370:   C806 20 B6 C7           JSR  CARET
5380:   C809 20 B6 C7           JSR  CARET
5390:   C80C 68                 PLA
5400:   C80D 20 26 C7           JSR  HEXASC
5410:   C810 20 A5 FF MELAUS    JSR  IECIN
5420:   C813 C9 0D              CMP  #CR
5430:   C815 F0 09              BEQ  MELEND
5440:   C817 28                 PLP
5450:   C818 08                 PHP
5460:   C819 90 F5              BCC  MELAUS
5470:   C81B 20 D2 FF           JSR  BASOUT
5480:   C81E 90 F0              BCC  MELAUS
5490:   C820 20 AB FF MELEND    JSR  UNTALK
5500:   C823 28                 PLP
5500:   C824 60                 RTS
5502:   C825 20 38 F8 TASTE     JSR  PTASTE
5502:   C828 20 CD C7           JSR  STOP2
5502:   C82B D0 F8              BNE  TASTE
5504:   C82D A9 07              LDA  #7
5504:   C82F 85 C0              STA  MOFLAG
5506:   C831 4C CA FC           JMP  MOTAUS
5511:   C834
5511:   C834
5511:   C834
                      ; AUSGABE STRINGS
5513:   C834
5513:   C834
5513:   C834
5520:   C834 0D 0D 0D QUEINS    .BYT 13,13,13
5530:   C837 12 51 55           .ASC "■QUELLDISK■ EINLEGEN !"
5540:   C84D 00                 .BYT 0
5550:   C84E 0D 0D    ZICASS    .BYT 13,13
5560:   C850 12 5A 49           .ASC "■ZIELCASSETTE■ EINLEGEN !"
5570:   C869 0D 00              .BYT 13,0
5580:   C86B 93 11 1C TITEL     .ASC "■■■      BACKUP VON DISKETTE AUF"
5590:   C88C 0D 0D              .BYT 13,13
5600:   C88E 1F 20 20           .ASC "■              CASSETTE"
5602:   C8A5 0D 0D              .BYT 13,13
5604:   C8A7 20 20 20           .ASC "         MIT ■■F A S T T A P E ■"
5606:   C8C6 0D 0D 0D           .BYT 13,13,13
5608:   C8C9 20 20 20           .ASC "   (C)    DIRK PAULISSEN  "
5610:   C8E3 0D 0D 0D           .BYT 13,13,13
5620:   C8E6 1E 20 20           .ASC "■          E = ENDE■"
5630:   C8FB 00                 .BYT 0
5640:   C8FC 0D       SLONG     .BYT 13
5650:   C8FD 1C 12 21           .ASC "■■!!■ ZU LANGE ZUM COPIEREN ■"
5660:   C91A 00                 .BYT 0
5670:   C91B 0D       LILONG    .BYT 13
5680:   C91C 1C 12 21           .ASC "■■!!■ LISTE IST VOLL ■"
5690:   C932 00                 .BYT 0
5700:   C933 12 4A 41 JANEIN    .ASC "■JA/NEIN■■■■■■■■■■"
5710:   C944 00                 .BYT 0
5720:   C945 12 1E 20 STRJA     .ASC "■■  JA  ■ ■"
5730:   C950 00                 .BYT 0
5740:   C951 0D       STRILL    .BYT 13
5750:   C952 1C 12 21           .ASC "■■!!■ ILLEGALER FILETYP■"
5760:   C96A 00                 .BYT 0
```

```
5770:   C96B 1C 12 20 STRNO    .ASC "▒▒ NEIN ▬ ▒"
5780:   C976 00                .BYT 0
5790:   C977 0D 0D 0D FRAGE    .BYT 13,13,13
5800:   C97A 57 4F 4C          .ASC "WOLLEN SIE DIE PROGRAMME"
5810:   C992 0D 0D             .BYT 13,13
5820:   C994 20 20 20          .ASC "    1     DURCHGEHEND"
5830:   C9A8 0D 0D             .BYT 13,13
5840:   C9AA 20 20 20          .ASC "    2     EINZELN"
5850:   C9BA 0D 0D             .BYT 13,13
5860:   C9BC 43 4F 50          .ASC "COPIEREN ?"
5870:   C9C6 00                .BYT 0
5880:   C9C7 0D 0D    SREAD    .BYT 13,13
5890:   C9C9 52 45 41          .ASC "READING "
5900:   C9D1 00                .BYT 0
5910:   C9D2 2C 50 2C PRGRE    .ASC ",P,R"
5920:   C9D6 00                .BYT 0
5930:   C9D7 55 49 2D UIMIN    .ASC "UI-"
5940:   C9DA 00                .BYT 0
5950:   C9DB 55 49 2B UIPLU    .ASC "UI+"
5960:   C9DE 00                .BYT 0
5970:   C9DF 49       STRI     .ASC "I"
5980:   C9E0 00                .BYT 0
5985:   C9E1 24       CATALO   .ASC "$"
5986:   C9E2 00                .BYT 0
5990:   C9E3 0D 0D    WRITI    .BYT 13,13
6000:   C9E5 53 41 56          .ASC "SAVING "
6010:   C9EC 00                .BYT 0
6020:   C9ED 0D 0D    EOCOP    .BYT 13,13
6030:   C9EF 1C 12 45          .ASC "▒▒ENDE DES COPIERENS ▬▒"
6040:   CA06 00                .BYT 0
6045:   CA07 0D       WMACH    .BYT 13
6050:   CA08 20 57 45          .ASC " WEITER MACHEN ? "
6060:   CA19 00                .BYT 0
6070:   CA1A 0D 0D    SNFILE   .BYT 13,13
6080:   CA1C 4E 41 45          .ASC "NAECHSTES FILE : "
6090:   CA2D 00                .BYT 0
```

```
***********************************************************
 100 REM BACKUP DISKETTE => CASSETTE FUER C-64
***********************************************************
 110 E=51757:A=50176:PS=0
 120 FOR I=A TO E:READ X:POKE I,X:PS=PS+X:NEXT
 130 IF PS<>171785 THEN PRINT"FEHLER IN DATAS":END
 140 SYS A:NEW
 150 DATA 169,1,141,32,208,141,33,208,169,6,141,134,2,169
     ,107,160,200,32,30,171
 160 DATA 169,52,160,200,32,30,171,32,153,199,201,69,208,
     1,96,32,182,199,32,182
 170 DATA 199,32,159,199,169,8,170,160,0,32,186,255,169,1,
     162,225,160,201,32
 180 DATA 189,255,32,192,255,169,8,32,180,255,169,0,32,150
     ,255,160,4,32,165,255
 190 DATA 136,208,250,32,165,255,133,34,32,165,255,166,34
     ,32,205,189,32,179,199
 200 DATA 32,165,255,240,6,32,210,255,24,144,245,32,182,1
     99,32,182,199,32,165
 210 DATA 255,32,165,255,160,0,132,251,32,165,255,133,252,
     32,165,255,133,253
 220 DATA 166,252,32,205,189,32,179,199,160,0,32,165,255,
     72,32,187,199,104,153
 230 DATA 64,3,240,3,200,208,240,32,165,255,32,165,255,16
     5,144,240,3,76,71,197
 240 DATA 165,253,208,6,165,252,201,152,144,10,169,252,16
     0,200,32,30,171,76,65
 250 DATA 197,165,251,201,48,144,10,169,27,160,201,32,30,
     171,24,144,237,169,0
 260 DATA 133,8,169,31,133,211,169,51,160,201,32,30,171,3
     2,225,199,201,78,240
 270 DATA 82,201,74,208,245,169,69,160,201,32,30,171,165,
     251,32,241,198,162,0
 280 DATA 232,189,64,3,201,34,208,248,134,253,232,189,64,
     3,201,34,240,6,145,65
 290 DATA 232,200,208,243,138,164,251,24,229,253,153,64,2
     02,189,64,3,208,10,169
 300 DATA 81,160,201,32,30,171,76,65,197,201,83,240,242,2
```

```
    01,80,240,3,232,208
310 DATA 230,230,251,24,144,7,169,107,160,201,32,30,171,
    32,182,199,76,125,196
320 DATA 169,8,32,195,255,165,251,208,3,76,0,196,169,119,
    160,201,32,30,171,32
330 DATA 225,199,201,49,240,7,201,50,208,245,169,0,44,169
    ,255,133,254,169,15
340 DATA 32,195,255,162,0,134,252,169,199,160,201,32,30,
    171,164,252,190,64,202
350 DATA 165,252,32,241,198,177,65,32,210,255,200,202,20
    8,247,169,2,162,8,168
360 DATA 32,186,255,166,252,189,64,202,133,34,165,252,32,
    241,198,162,0,177,65
370 DATA 157,64,3,200,232,198,34,208,245,160,0,185,210,2
    01,157,64,3,200,232
380 DATA 192,4,144,244,138,162,64,160,3,32,189,255,32,19
    2,255,169,9,160,0,132
390 DATA 34,133,35,173,17,208,41,239,141,17,208,169,15,1
    62,8,168,32,186,255
400 DATA 169,3,162,215,160,201,32,189,255,32,192,255,162,
    2,32,15,243,32,31,243
410 DATA 165,186,32,180,255,165,185,32,150,255,160,0,32,
    165,255,32,9,199,166
420 DATA 144,240,246,32,235,199,8,169,2,32,195,255,173,1
    7,208,9,16,141,17,208
430 DATA 40,144,3,32,55,199,166,252,165,34,133,174,165,3
    5,133,175,232,165,252
440 DATA 208,13,169,78,160,200,32,30,171,32,225,199,32,3
    7,200,169,227,160,201
450 DATA 32,30,171,164,252,190,64,202,165,252,32,241,198,
    177,65,32,210,255,200
460 DATA 202,208,247,162,1,169,0,168,32,186,255,166,252,
    189,64,202,133,34,165
470 DATA 252,32,241,198,162,0,177,65,157,64,3,232,200,19
    8,34,208,245,138,162
480 DATA 64,160,3,32,189,255,160,0,169,9,132,172,133,173,
    177,172,133,167,200
490 DATA 177,172,133,168,200,132,172,165,174,56,229,172,8
    ,24,101,167,133,169
500 DATA 165,175,101,168,40,229,173,133,170,162,5,134,171
```

```
,32,133,192,32,229
510 DATA 198,230,252,166,252,228,251,176,7,36,254,16,16,
    76,223,198,169,237,160
520 DATA 201,32,30,171,32,225,199,76,0,196,32,85,199,169,
    78,160,200,32,30,171
530 DATA 32,225,199,32,37,200,32,182,199,76,117,197,32,1
    15,199,173,17,208,9
540 DATA 16,141,17,208,96,160,0,10,10,132,66,10,38,66,10,
    38,66,133,65,165,66
550 DATA 24,105,203,133,66,160,0,96,145,34,230,34,208,2,
    230,35,88,96,32,165
560 DATA 255,41,15,10,10,10,10,133,87,32,165,255,41,15,5,
    87,96,72,74,74,74,74
570 DATA 32,47,199,104,41,15,24,105,48,76,210,255,173,17
    ,208,9,16,141,17,208
580 DATA 169,7,160,202,32,30,171,32,225,199,201,89,208,1,
    96,201,78,208,244,76
590 DATA 208,199,169,26,160,202,32,30,171,165,252,10,10,
    10,10,166,252,188,64
600 DATA 202,170,189,0,203,32,210,255,232,136,208,246,96,
    173,17,208,41,239,141
610 DATA 17,208,169,15,32,195,255,169,15,162,8,168,32,186
    ,255,169,3,162,219
620 DATA 160,201,32,189,255,32,192,255,169,15,76,195,255
    ,32,228,255,240,251
630 DATA 96,169,15,162,8,168,32,186,255,169,1,162,223,16
    0,201,32,189,255,76
640 DATA 192,255,169,32,44,169,13,76,210,255,32,210,255,
    138,72,152,72,32,225
650 DATA 255,24,208,1,56,104,168,104,170,176,1,96,162,24
    6,154,32,229,198,169
660 DATA 15,32,195,255,32,231,255,76,0,196,32,228,255,20
    8,233,32,190,199,144
670 DATA 246,162,15,32,15,243,32,31,243,165,186,32,180,2
    55,165,185,32,150,255
680 DATA 32,19,199,201,32,8,144,11,72,32,182,199,32,182,
    199,104,32,38,199,32
690 DATA 165,255,201,13,240,9,40,8,144,245,32,210,255,14
    4,240,32,171,255,40
700 DATA 96,32,56,248,32,205,199,208,248,169,7,133,192,7
```

```
    6,202,252,13,13,13,18
710 DATA 81,85,69,76,76,68,73,83,75,146,32,69,73,78,76,6
    9,71,69,78,32,33,0,13
720 DATA 13,18,90,73,69,76,67,65,83,83,69,84,84,69,146,3
    2,69,73,78,76,69,71
730 DATA 69,78,32,33,13,0,147,17,28,32,32,32,32,32,32,32,
    66,65,67,75,85,80,32
740 DATA 86,79,78,32,68,73,83,75,69,84,84,69,32,65,85,70
    ,13,13,31,32,32,32,32
750 DATA 32,32,32,32,32,32,32,32,32,32,67,65,83,83,69,84,
    84,69,13,13,32,32,32
760 DATA 32,32,32,32,32,77,73,84,32,18,129,70,32,65,32,83
    ,32,84,32,84,32,65
770 DATA 32,80,32,69,32,146,13,13,13,32,32,32,40,67,41,32
    ,32,32,32,68,73,82
780 DATA 75,32,80,65,85,76,73,83,83,69,78,32,32,13,13,13
    ,30,32,32,32,32,32,32
790 DATA 32,32,32,32,32,69,32,61,32,69,78,68,69,31,0,13,
    28,18,33,33,18,32,90
800 DATA 85,32,76,65,78,71,69,32,90,85,77,32,67,79,80,73,
    69,82,69,78,32,31,0
810 DATA 13,28,18,33,33,18,32,76,73,83,84,69,32,73,83,84
    ,32,86,79,76,76,32,31
820 DATA 0,18,74,65,47,78,69,73,78,157,157,157,157,157,1
    57,157,146,31,0,18,30
830 DATA 32,32,74,65,32,32,146,32,31,0,13,28,18,33,33,18,
    32,73,76,76,69,71,65
840 DATA 76,69,82,32,70,73,76,69,84,89,80,31,0,28,18,32,
    78,69,73,78,32,146,32
850 DATA 31,0,13,13,13,87,79,76,76,69,78,32,83,73,69,32,
    68,73,69,32,80,82,79
860 DATA 71,82,65,77,77,69,13,13,32,32,32,32,49,32,32,32,
    32,68,85,82,67,72,71
870 DATA 69,72,69,78,68,13,13,32,32,32,32,50,32,32,32,32
    ,69,73,78,90,69,76,78
880 DATA 13,13,67,79,80,73,69,82,69,78,32,63,0,13,13,82,
    69,65,68,73,78,71,32
890 DATA 0,44,80,44,82,0,85,73,45,0,85,73,43,0,73,0,36,0,
    13,13,83,65,86,73,78
900 DATA 71,32,0,13,13,28,18,69,78,68,69,32,68,69,83,32,
```

```
    67,79,80,73,69,82,69
910 DATA 78,83,32,146,31,0,13,32,87,69,73,84,69,82,32,77,
    65,67,72,69,78,32,63
920 DATA 32,0,13,13,78,65,69,67,72,83,84,69,83,32,70,73,
    76,69,32,58,32,0
```

```
**********************************************************
 100 REM BACKUP DISKETTE => CASSETTE FUER VC20
**********************************************************
 110 E=31156:A=29696:PS=0
 120 FOR I=A TO E:READ X:POKE I,X:PS=PS+X:NEXT
 130 IF PS<>145927 THEN PRINT"FEHLER IN DATAS":END
 140 SYS A:NEW
 150 DATA 169,25,141,15,144,169,6,141,134,2,169,19,160,12
     0,32,30,203,169,221
 160 DATA 160,119,32,30,203,32,69,119,201,69,208,1,96,32,
     98,119,32,98,119,32
 170 DATA 75,119,169,8,170,160,0,32,186,255,169,1,162,103,
     160,121,32,189,255
 180 DATA 32,192,255,169,8,32,180,255,169,0,32,150,255,160
     ,4,32,165,255,136,208
 190 DATA 250,32,165,255,133,34,32,165,255,166,34,32,205,
     221,32,95,119,32,165
 200 DATA 255,240,6,32,210,255,24,144,245,32,98,119,32,98,
     119,32,165,255,32,165
 210 DATA 255,160,0,132,3,32,165,255,133,4,32,165,255,133
     ,5,166,4,32,205,221
 220 DATA 32,95,119,160,0,32,165,255,72,32,103,119,104,15
     3,64,3,240,3,200,208
 230 DATA 240,32,165,255,32,165,255,165,144,240,3,76,64,1
     17,165,5,208,6,165,4
 240 DATA 201,93,144,10,169,133,160,120,32,30,203,76,58,1
     17,165,3,201,48,144
 250 DATA 10,169,164,160,120,32,30,203,24,144,237,169,0,1
     33,8,169,188,160,120
 260 DATA 32,30,203,32,138,119,201,78,240,82,201,74,208,2
     45,169,206,160,120,32
 270 DATA 30,203,165,3,32,203,118,162,0,232,189,64,3,201,
     34,208,248,134,5,232
 280 DATA 189,64,3,201,34,240,6,145,65,232,200,208,243,13
     8,164,3,24,229,5,153
 290 DATA 64,122,189,64,3,208,10,169,218,160,120,32,30,20
     3,76,58,117,201,83,240
 300 DATA 242,201,80,240,3,232,208,230,230,3,24,144,7,169,
```

```
    244,160,120,32,30,203
310 DATA 32,98,119,76,122,116,169,8,32,195,255,165,3,208
    ,3,76,0,116,169,0,160
320 DATA 121,32,30,203,32,138,119,201,49,240,7,201,50,20
    8,245,169,0,44,169,255
330 DATA 133,5,169,15,32,195,255,162,0,134,4,169,77,160,
    121,32,30,203,164,4
340 DATA 190,64,122,165,4,32,203,118,177,65,32,210,255,2
    00,202,208,247,169,2
350 DATA 162,8,168,32,186,255,166,4,189,64,122,133,34,16
    5,4,32,203,118,162,0
360 DATA 177,65,157,64,3,200,232,198,34,208,245,160,0,18
    5,88,121,157,64,3,200
370 DATA 232,192,4,144,244,138,162,64,160,3,32,189,255,3
    2,192,255,169,19,160
380 DATA 0,132,34,133,35,169,15,162,8,168,32,186,255,169,
    3,162,93,160,121,32
390 DATA 189,255,32,192,255,162,2,32,207,243,32,223,243,
    165,186,32,180,255,165
400 DATA 185,32,150,255,160,0,32,165,255,32,227,118,166,
    144,240,246,32,148,119
410 DATA 8,169,2,32,195,255,40,144,3,32,17,119,166,4,165,
    34,133,174,165,35,133
420 DATA 175,232,165,4,208,13,169,247,160,119,32,30,203,
    32,138,119,32,206,119
430 DATA 169,105,160,121,32,30,203,164,4,190,64,122,165,
    4,32,203,118,177,65
440 DATA 32,210,255,200,202,208,247,162,1,169,0,168,32,1
    86,255,166,4,189,64
450 DATA 122,133,34,165,4,32,203,118,162,0,177,65,157,64,
    3,232,200,198,34,208
460 DATA 245,138,162,64,160,3,32,189,255,160,0,169,19,132
    ,172,133,173,177,172
470 DATA 133,167,200,177,172,133,168,200,132,172,165,174
    ,56,229,172,8,24,101
480 DATA 167,133,169,165,175,101,168,40,229,173,133,170,
    162,5,134,171,32,127
490 DATA 112,230,4,166,4,228,3,176,7,36,5,16,16,76,197,1
    18,169,115,160,121,32
500 DATA 30,203,32,138,119,76,0,116,32,39,119,169,247,16
```

```
     0,119,32,30,203,32,138
510 DATA 119,32,206,119,32,98,119,76,110,117,160,0,10,10,
     132,66,10,38,66,10
520 DATA 38,66,133,65,165,66,24,105,123,133,66,160,0,96,
     145,34,230,34,208,2
530 DATA 230,35,88,96,32,165,255,41,15,10,10,10,10,133,8
     7,32,165,255,41,15,5
540 DATA 87,96,72,74,74,74,74,32,9,119,104,41,15,24,105,
     48,76,210,255,169,141
550 DATA 160,121,32,30,203,32,138,119,201,89,208,1,96,20
     1,78,208,244,76,124
560 DATA 119,169,160,160,121,32,30,203,165,4,10,10,10,10,
     166,4,188,64,122,170
570 DATA 189,0,123,32,210,255,232,136,208,246,96,32,228,
     255,240,251,96,169,15
580 DATA 162,8,168,32,186,255,169,1,162,101,160,121,32,1
     89,255,76,192,255,169
590 DATA 32,44,169,13,76,210,255,32,210,255,138,72,152,7
     2,32,225,255,24,208
600 DATA 1,56,104,168,104,170,176,1,96,162,246,154,169,1
     5,32,195,255,32,231
610 DATA 255,76,0,116,32,228,255,208,236,32,106,119,144,
     246,162,15,32,207,243
620 DATA 32,223,243,165,186,32,180,255,165,185,32,150,25
     5,32,237,118,201,32
630 DATA 8,144,11,72,32,98,119,32,98,119,104,32,0,119,32,
     165,255,201,13,240
640 DATA 9,40,8,144,245,32,210,255,144,240,32,171,255,40
     ,96,32,183,248,32,121
650 DATA 119,208,248,169,7,133,192,76,8,253,13,13,13,18,
     81,85,69,76,76,68,73
660 DATA 83,75,146,32,69,73,78,76,69,71,69,78,32,33,0,13,
     13,18,90,73,69,76,67
670 DATA 65,83,83,69,84,84,69,146,32,69,73,78,76,69,71,69
     ,78,33,13,0,147,17
680 DATA 28,32,32,66,65,67,75,85,80,32,86,79,78,32,68,73
     ,83,75,69,84,84,69,13
690 DATA 13,31,32,32,32,32,32,32,65,85,70,32,67,65,83,83,
     69,84,84,69,13,13,77
700 DATA 73,84,32,18,129,70,32,65,32,83,32,84,32,84,32,65
```

```
    ,32,80,32,69,32,146
710 DATA 13,13,13,32,40,67,41,32,32,32,32,68,73,82,75,32
    ,80,65,85,76,73,83,83
720 DATA 69,78,13,13,13,30,32,32,32,32,32,69,32,61,32,69,
    78,68,69,31,0,13,28
730 DATA 18,33,33,18,32,90,85,32,76,65,78,71,69,32,90,85
    ,77,32,67,79,80,73,69
740 DATA 82,69,78,32,31,0,13,28,18,33,33,18,32,76,73,83,
    84,69,32,73,83,84,32
750 DATA 86,79,76,76,32,31,0,18,74,65,47,78,69,73,78,157,
    157,157,157,157,157
760 DATA 157,146,31,0,18,30,32,32,74,65,32,32,146,32,31,0
    ,13,28,18,33,33,18
770 DATA 32,73,76,76,69,71,65,76,69,82,32,70,73,76,69,84
    ,89,80,31,0,28,18,32
780 DATA 78,69,73,78,32,146,32,31,0,13,13,13,87,79,76,76,
    69,78,32,83,73,69,32
790 DATA 68,73,69,32,13,80,82,79,71,82,65,77,77,69,13,13
    ,32,32,49,32,32,32,32
800 DATA 68,85,82,67,72,71,69,72,69,78,68,13,13,32,32,50,
    32,32,32,32,69,73,78
810 DATA 90,69,76,78,13,13,67,79,80,73,69,82,69,78,32,63
    ,0,13,13,82,69,65,68
820 DATA 73,78,71,32,0,44,80,44,82,0,85,73,45,0,85,73,43,
    0,73,0,36,0,13,13,83
830 DATA 65,86,73,78,71,32,0,13,13,28,18,69,78,68,69,32,
    68,69,83,32,67,79,80
840 DATA 73,69,82,69,78,83,32,146,31,0,13,32,87,69,73,84,
    69,82,32,77,65,67,72
850 DATA 69,78,32,63,32,0,13,13,78,65,69,67,72,83,84,69,
    83,32,70,73,76,69,32
860 DATA 58,32,13,00
```

15.4 PROGRAMMBESCHREIBUNG BACKUP DISK-CC

Für Interessierte und der ASSEMBLER-Sprache Kundige nun noch eine Beschreibung des nicht gerade kurzen Programms.

Vorbemerkung:

Dieses Programm arbeitet mit zwei unterschiedlichen Lade- bzw Speicherstartadressen. Die tatsächliche Startadresse ist die Adresse des Puffers, unter der das Programm zum kopieren zwischengespeichert wird. Die eigentliche Startadresse ist die, unter der das Programm normalerweise steht und arbeitet.

Das Programmprinzip ist folgendes:

Nach der Disketteninitialisierung werden zuerst die Filenamen eingelesen und im Cassettenpuffer ab $0340 zwischengespeichert. Dann wird der Anwender gefragt, ob das Programm kopiert werden soll oder nicht. Wird die Frage mit "J" beantwortet, wird getestet, ob die maximale Blocklänge bzw. die Filenamenlistenlänge überschritten wird und ggf. eine Fehlermeldung ausgegeben. Wird keine Überschreitung festgestellt, wird der Filename und seine Länge in zwei Tabellen (NAMTAB, LNGTAB) hinter dem Backup-Programm übertragen. Weiterhin testet das Programm auch auf den Filetyp des zu kopierenden Files. Handelt es sich nicht um ein Programmfile, wird eine Fehlermeldung ausgelesen.

Wurde auf diese Weise das ganze Inhaltsverzeichnis bearbeitet, wird in die eigentliche Copierroutine gesprungen.

Nach Ausgabe von "READING name" wird der entsprechende Filename wieder in den Cassettenpuffer übertragen und ",P,R" angehängt. Mit dem so erweiterten Filenamen wird eine Lesedatei eröffnet und das entsprechende Programm nach $0900 (VC 20 = $1300), der tatsächlichen Programmladeadresse (TLSADR), geladen. Wenn beim Lesen kein Fehler aufgetreten ist, wird der FastTape-Speicherroutine die Programmstart-

und Endadresse, die in den Programm-Header geschrieben werden soll und die tatsächliche Programmstart- und Endadresse, unter der das Programm zur Zeit abgespeichert wurde, übergeben. Danach wird zur FastTape-Routine "ABSOLUT" verzweigt. Anschließend wird wieder zum Start zurückgesprungen, falls noch weitere Files kopiert werden sollen.

Ich habe das ASSEMBLER-Listing mit vielen Kommentaren versehen, um den Programmablauf weitgehendst klar zu machen. Zum Abschluß noch eine grundsätzliche Bemerkung.

Die Diskettenstation kann mit zwei Übertragungsgeschwindigkeiten arbeiten. Eine schnellere, speziell für den VC 20, und eine langsame für den C-64, da dieser durch die Bildschirmsteuerung etwas langsamer arbeitet. Schaltet man den Bildschirm aus, arbeitet der C-64 genauso schnell wie der VC 20 und kann so mit der höheren Übertragungsrate die Diskette ansteuern. Aus diesem Grunde wird bei dem Ladevorgang der Bildschirm ausgeschaltet und die Diskette auf die höhere Übertragungsrate umgestellt.

Das erklärt, warum für den VC 20 die entsprechenden Umschaltungen, die im Listing deutlich gekennzeichnet sind, entfallen.

A N H A N G

A WICHTIGE SPEICHERSTELLEN

Hex	Dezimal	Bedeutung
+0001	1	Ausgaberegister der CPU
		Bit3 Schreibleitung für Datassette
		Bit4 CASS-SENS, Recordertastenabf.
		Bit5 Motorsteuerung
000A	10	LOAD/VERIFY - Flag; 0=LOAD/1=VERIFY
002B-002C	33 - 34	Zeiger Basic-RAM-Start
002D-002E	35 - 36	Zeiger Basicprogramm Ende/Variablen Start
002F-0030	37 - 38	Zeiger Beginn der Felder
0031-0032	49 - 50	Zeiger Ende der Felder
0033-0034	51 - 52	Zeiger auf Ende der Strings
0037-0038	55 - 56	Zeiger auf Ende des Basic-RAM
0090	144	Statusbyte
0093	147	LOAD/VERIFY - Flag; 0=LOAD/1=VERIFY
0098	152	Anzahl der offenen Files
0099	153	Eingabegerät
009A	154	Ausgabegerät
00A6	166	Zeiger in Cassettenpuffer
00AC-00AD	172 - 173	Zeiger auf aktuelles Byte beim Schreiben und Lesen
00AE-00AF	174 - 175	Zeiger auf Prg. Ende beim Lesen/Schreiben
00B2-00B3	178 - 179	Zeiger auf Start des Cassettenpuffers
00B7	183	Filenamenlänge
00B8	184	Aktuelle logische Filenummer
00B9	185	Aktuelle Sekundäradresse
00BA	186	Aktuelle Geräteadresse
00BE	190	Pass-Zähler beim Lesen und Schreiben
00BD	191	Ein-/Ausgabebyte
00C0	192	Cassettenmotor Flag
00C1-00C2	193 - 194	Ein-/Ausgabe Startadresse
00C3-00C4	195 - 196	Ein-/Ausgabe Endadresse
033C-03FB	828 - 1019	Cassettenpuffer

033C	828		Filetyp
033D-033E	829 -	830	Prg. Startadresse
033F-0340	831 -	832	Prg. Endadresse
0341-	833 -		Filenamen
*911c	37148		Bit5 Cassettenmotor Schalter
*911F	37151		Bit6 CASS-SENS; Recordertastenabfr.
*9120	37152		Bit3 Schreibleitung

+ : Gilt nur für C-64

* : Gilt nur für VC 20

Laden, Starten – Klar!



DIE DISKETTE ZUM BUCH

Über 500.000 DATA BECKER Bücher sind bisher verkauft und das nicht ohne Grund. Die beliebten DATA BECKER Standardwerke zum VC 20 und zum COMMODORE 64 stecken voller Programmiertips und Listings, die jeder Leser am liebsten sofort am Gerät ausprobieren möchte. Doch ohne fleißiges Abtippen der Programme läuft nichts. Abtippen ist so langweilig wie unzuverlässig; der kleinste Tippfehler kann den ganzen Spaß verderben. Ab sofort nimmt Ihnen DATA BECKER diese Arbeit ab. Die DISKETTEN ZUM BUCH enthalten alle Programme und Utilities, die Sie als Listing im jeweiligen Buch finden. Diskette ins Laufwerk, gewünschtes Programm laden, starten und schon können Sie mit der ausgetüftelten Software der DATA BECKER Autoren arbeiten. Und: Sie haben die Sicherheit, daß diese Programme wirklich auf Anhieb laufen. Ist das nichts?

Die Diskette zum Buch:
Das große Drucker-Buch. DM 39,–

Die Diskette zum Buch:
Das Maschinensprachebuch zum COMMODORE 64. DM 39,–

Die Diskette zum Buch:
Das Maschinensprachebuch für Fortgeschrittene zum COMMODORE 64. DM 39,–

Die Diskette zum Buch:
Das große Floppy-Buch. DM 39,–

Die Diskette zum Buch:
64 Tips & Tricks. DM 39,–

Die Diskette zum Buch: Das Schulbuch zum COMMODORE 64.
Diese Superdiskette enthält zusätzlich noch 14 weitere Lernprogramme und ein ca. 70seitiges Handbuch. Ein absolutes Muß für Schüler, Lehrer und Eltern. DM 49,–

Neu Die Diskette zum Buch:
Pascal 64 Tips & Tricks
DM 39,–

Die Diskette zum Buch:
Das Grafikbuch zum COMMODORE 64. DM 39,–

Die Diskette zum Buch:
64 INTERN. DM 39,–

Die Diskette zum Buch:
64 für Profis. DM 39,–

Die Diskette zum Buch:
VC-20 Tips & Tricks. DM 39,–

Die Diskette zum Buch:
APPLE II Tips & Tricks. DM 39,–

Neu Die Diskette zum Buch:
Das Trainingsbuch zu Datamat
DM 39,–

Neu Die Diskette zum Buch:
Adventures – und wie man sie programmiert
DM 49,–

DATA BECKER BÜCHER & PROGRAMME erhalten Sie im Computer-Fachhandel, in guten Buchhandlungen und in den Fachabteilungen der Kauf- und Warenhäuser.

DATA BECKER

Merowingerstr. 30 · 4000 Düsseldorf · Tel. (0211) 310010 · im Hause AUTO BECKER

DATA BECKER'S NEUE BÜCHER UND PROGRAMME FÜR COMMODORE

Spickzettel ade.



Ein neues DATA BECKER BUCH, das den Einsatz des COMMODORE 64 in der Schule entscheidend mitprägen dürfte, wurde von Professor Voß geschrieben. Besonders für Schüler der Mittel- und Oberstufe geschrieben, enthält das Buch viele interessante Problemlösungs- und Lernprogramme, die besonders ausführlich und leicht verständlich beschrieben sind. Sie ermöglichen ein intensives und anregendes Lernen, unter anderem mit folgenden Themen: Satz des Pythagoras, quadratische Gleichungen, geometrische Reihen, Pendelbewegungen, mechanische Hebel, Molekülbildung, exponentielles Wachstum, Vokabeln lernen, unregelmäßige Verben, Zinseszinsrechnung. Ein kurzer Überblick über die Grundlagen der EDV, eine knappe Wiederholung der wichtigsten BASIC-Elemente und eine Einführung in die Grundzüge der Problemanalyse vervollständigen das Ganze. Mit diesem Buch machen die Hausaufgaben wieder Spaß!

DAS SCHULBUCH ZUM COMMODORE 64, 1984, über 300 Seiten, DM 49,–

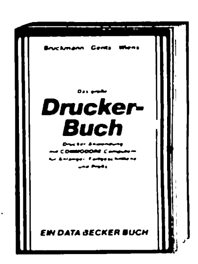
Tempo!



MASCHINENSPRACHE FÜR FORTGESCHRITTENE ist bereits das zweite Buch von Lothar Englisch zum Thema Maschinenprogrammierung mit dem COMMODORE 64. Hier wird von der Problemanalyse bis zum Maschinensprachealgorithmus in die Grundlagen der professionellen Maschinenspracheprogrammierung eingeführt. In diesem Buch finden Sie unter anderem folgende Themen behandelt: Problemlösungen in Maschinensprache, Programmierung von Interruptroutinen, Interruptquellen beim COMMODORE 64, Interrupts durch CIA's und Videocontroller, Programmierung der Ein-Ausgabe-Bausteine, die CIA's des COMMODORE 64, Timer, Echtzeituhr, parallele und serielle Ein/Ausgabe, BASIC-Erweiterungen, Programmierung eigener BASIC-Befehle und -Funktionen, Möglichkeiten zur Einbindung ins Betriebssystem sowie viele weitere Tips & Tricks zur Maschinenprogrammierung. Dieses Buch sollte jeder haben, der wirklich intensiv mit der Maschinensprache des COMMODORE 64 arbeiten will.

MASCHINENSPRACHE FÜR FORTGESCHRITTENE, 1984, ca. 200 Seiten, DM 39,–

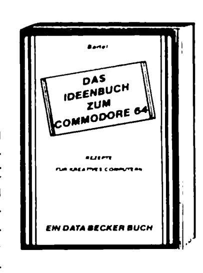
Macht Druck.



DAS GROSSE DRUCKERBUCH für Drucker-Anwender mit COMMODORE-Computern ist endlich da! Es enthält eine riesige Sammlung von Tips & Tricks, Programmlistings und Hardwareinformationen. Rolf Brückmann und Klaus Gerits beschäftigen sich mit Sekundäradressen, Anschluß einer Schreibmaschine am Userport, Druckerschnittstellen (Centronics, V 24, IEC-Bus), hochauflösender Grafik, Text- und Grafikhardcopy, Grafik mit Standardzeichensatz, formatierter Datenausgabe, Plakatschrift, Textverarbeitung und vieles mehr. Zusätzlich wird das Betriebssystem des MPS801 zerlegt, mit Prozessorbeschreibung (8035), Blockschaltbild und einem ausführlich kommentierten ROM-Listing. Thomas Wiens schrieb den Teil über die Programmierung des Plotters VC-1520: Handhabung des Plotters, Programmierung von Sonderzeichen, Funktionendarstellung, Kuchen und Säulendiagramme, Entwurf dreidimensionaler Gegenstände. Natürlich wieder viele interessante Listings. Unentbehrlich für jeden, der einen COMMODORE 64 oder VC-20 und einen Drucker besitzt.

DAS GROSSE DRUCKERBUCH, 1984, über 300 Seiten, DM 49,–

Tausendsassa.



Fast alles, was man mit dem COMMODORE 64 machen kann, ist in diesem Buch ausführlich beschrieben. Es ist nicht nur spannend zu lesen wie ein Roman, sondern enthält neben nützlichen Programmlistings vor allem viele, viele Anwendungsmöglichkeiten des C64. Dabei wurde besonderer Wert darauf gelegt, daß das Buch auch für Laien leicht verständlich ist. Eine Auswahl aus der Themenvielfalt: Gedichte vom Computer, Einladung zur Party, Diplomarbeit – professionell gestaltet, individuelle Werbebriefe, Autokosten im Griff, Baukostenberechnung, Taschenrechner, Rezeptkartei, Lagerliste, persönliches Gesundheitsarchiv, Diätplan elektronisch, intelligentes Wörterbuch, kleine Notenschule, CAD für Handarbeit, Routenoptimierung, Schaufensterwerbung, Strategiespiele. Teilweise sind Programmlistings fertig zum Eintippen enthalten, soweit sich die „Rezepte" auf 1–2 Seiten realisieren ließen. Wenn Sie bisher nicht immer wußten, was Sie mit Ihrem 64er alles anfangen sollten, nach dem Lesen des IDEENBUCHES wissen Sie's bestimmt!

DAS IDEENBUCH ZUM COMMODORE 64, 1984, über 200 Seiten, DM 29,–

Prof. 64.



Ein faszinierendes Buch, um in die Welt der Wissenschaft einzusteigen, hat Rainer Severin geschrieben. Zunächst werden Variablentypen, Rechengenauigkeit und nützliche POKE-Adressen des COMMODORE 64 bezüglich den Anforderungen wissenschaftlicher Probleme analysiert. Verschiedene Sortieralgorithmen wie Bubble, Quick und Shell-Sort werden miteinander verglichen. Die Programmbeispiele aus der Mathematik nehmen dabei eine zentrale Stelle im Buch ein: Nullstellen nach Newton, numerische Ableitung mit dem Differenzenquotienten, lineare und nichtlineare Regression, Chi-Quadrat-Verteilung und Anpassungstest, Fourieranalyse und -synthese, Skalar-, Vektor- und Spatprodukt, ein Programmpaket zur Matrizenrechnung für Inversion, Eigenwerte und vieles weitere mehr. Programme aus der Chemie (Periodensystem), Physik, Biologie (Schadstoffe in Gewässern – Erfassung der Meßwerte), Astronomie (Planetenpositionen) und Technik (Berechnung komplexer Netzwerke, Platinenlayout am Bildschirm) und viele weitere Softwarelistings zeigen die riesigen Möglichkeiten auf, die der Computer in Wissenschaft und Technik hat.

COMMODORE 64 FÜR TECHNIK UND WISSENSCHAFT, 1984, über 200 Seiten, DM 49,–
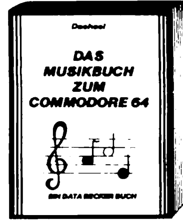
Grundkurs.



Das neue BASIC-Trainingsbuch zum C-64 ist eine ausführliche, didaktisch gut geschriebene Einführung in das CBM BASIC V2. Alle Befehle werden ausführlich erläutert. Dieses Buch geht aber über eine reine Befehlsbeschreibung hinaus, es wird eine fundierte Einführung in die Programmierung gegeben. Von der Problemanalyse bis zum fertigen Algorithmus lernt man das Entwerfen eines Programmes und den Entwurf von Datenflußplänen. ASCII-Code und verschiedene Zahlensysteme wie hexadezimal, binär und dezimal sind nach der Lektüre des Buches keine Fremdworte mehr. Die Programmierung von Schleifen, Sprüngen, bedingten Sprüngen lernt man leicht durch „learning by doing". So enthält das Trainingsbuch viele Aufgaben, Übungen und unzählige Beispiele. Den Schluß des Buches bildet eine Einführung ins professionelle Programmieren, in der es um mehrdimensionale Felder, Menuesteuerung und Unterprogrammtechnik geht. Endlich ein Buch, das Ihnen wirklich hilft, solide und sicher BASIC zu lernen.

BASIC TRAININGSBUCH ZUM COMMODORE 64, 1984, ca. 250 Seiten, DM 39,–

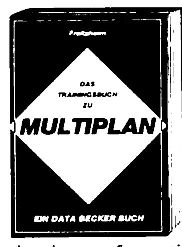
Sang und Klang!



Der COMMODORE 64 ist ein Musikgenie. DAS MUSIKBUCH hilft Ihnen, die riesigen Klangmöglichkeiten des C64 zu nutzen. Die Themenbreite reicht von einer Einführung in die Computermusik über die Erklärung der Hardwaregrundlagen des COMMODORE 64 und die Programmierung in BASIC bis hin zur fortgeschrittenen Musikprogrammierung in Maschinensprache. Einiges aus dem Inhalt: Soundregister des COMMODORE 64, Gate-Signal, Programmierung der "ADSR"-Werte, Synchronisation und Ring-Modulation, Counterprinzip, lineare und nichtlineare Musikprogrammierung, Frequenzmodulation, Interrupts in der Musikprogrammierung und vieles mehr. Zahlreiche Beispielprogramme, komplette Songs und nützliche Routinen ergänzen den Text. Geschrieben wurde das Buch von Thomas Dachsel, dem Autor der weltbekannten Musikprogramme Synthimat und Synthesound. Erschließen Sie sich die Welt des Sounds und der Computermusik mit dem Musikbuch zum C-64!

DAS MUSIKBUCH ZUM COMMODORE 64, über 200 Seiten, DM 39,–

Nützlich.



Das Trainingsbuch zu MULTIPLAN bietet eine gute Einführung in die Grundlagen der Tabellenkalkulation. Dabei wird großer Wert auf ein möglichst schnelles Einarbeiten in die wichtigsten Befehle gelegt, so daß man bald sicher mit MULTIPLAN arbeiten kann, ob nun auf dem COMMODORE 64 oder einem anderen Rechner. Am Ende wird man in der Lage sein, den umfangreichen Befehlssatz von MULTIPLAN auch kommerziell zu nutzen. Übungen am Ende jedes Kapitels sorgen dafür, daß man das Gelernte lange behält. Grundlage des Buches sind viele Seminare, die der Autor zu MULTIPLAN konzipiert und erfolgreich durchgeführt hat.

DAS TRAININGSBUCH ZU MULTIPLAN, 1984, ca. 250 Seiten, DM 49,–

Für Tüftler.



Ein hochinteressantes Buch für Hobbyelektroniker hat Rolf Brückmann vorgelegt. Er ist ein engagierter Techniker, für den der Computer Hobby und Beruf zur gleichen Zeit ist. Vor allem aber kennt er den C-64 in- und auswendig. So werden einführend die Schnittstellen des COMMODORE 64 detailliert beschrieben und kurz die Funktionsweise der CIAs 6526 erläutert. Hauptteil des Buches sind die Beschreibungen der vielfältigen Einsatzmöglichkeiten des COMMODORE 64. Die vielen Schaltungen, von Rolf Brückmann alle selbst

entwickelt, sind jeweils umfangreich dokumentiert und leichtverständlich erklärt. Die Reihe der hier ausführlich behandelten Anwendungen mit dem COMMODORE 64 ist äußerst umfangreich: Motorsteuerung, Stoppuhr mit Lichtschranke, Lichtorgel, A/D-Wandler, Spannungsmessung, Temperaturmessung und vieles mehr. Dazu kommen noch eine Reihe kompletter Schaltungen zum Selberbauen, wie ein EPROM Programmiergerät für den C-64, eine EPROM-Karte, ein Frequenzzähler und Sprachein/ausgabe (!). Zusätzlich sind jeweils Schaltplan, Softwarelisting und zu einigen Schaltungen sogar zusätzlich Platinenlayouts vorhanden.

DER COMMODORE 64 UND DER REST DER WELT, 1984, ca. 220 Seiten, DM 49,–

Computerkünstler.

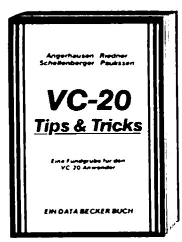
Das Grafikbuch zum COMMODORE 64 Buch aus der Bestseller-Serie von DATA BECKER stammt aus der Feder von Axel Plenge. Es geht weit über die reine Hardware-Beschreibung der Grafikeigenschaften des C-64 hinaus. Der Inhalt reicht von den Grundlagen der Grafikprogrammierung bis zum Computer Aided Design. Es ist ein Buch für alle, die mit ihrem C-64 kreativ tätig sein wollen. Themen sind z.B.: Zeichensatzprogrammierung, bewegte Sprites, High-Resolution, Multicolor-Grafik, Lightpenanwendungen, Betriebsarten des VIC, Verschieben der Bildschirmspeicher, IRQ-Handhabung, 3-Dimensionale Grafik, Projektionen, Kurven-, Balken- und Kuchendiagramme, Laufschriften, Animation, bewegte Bilder. Viele Programmlistings und Beispiele sind selbstverständlich. Das COMMODORE-BASIC V2 unterstützt die herausragenden Grafikeigenschaften des C-64 bekanntlich kaum. Hier helfen die vielen Beispielprogramme in diesem Buch weiter, die die faszinierende Welt der Computergrafik jedermann zugänglich machen. Kompetent ist der Autor dazu wie kaum ein anderer, schließlich hat er das äußerst leistungsfähige Programm SUPERGRAFIK geschrieben.



DAS GRAFIKBUCH ZUM COMMODORE 64, 1984, 295 Seiten, DM 39,–

Vielfalt.

Auf dem neuesten Stand ist VC-20 TIPS & TRICKS von Dirk Paulissen gebracht worden, der über hundert Seiten hinzufügte. Bisher schon enthalten waren Informationen über Speicheraufbau des VC-20 und die Erweiterungsmöglichkeiten, ein Grafikkapitel über programmierbare Zeichen, Laufschrift und die Supererweiterung. Stark erweitert wurde der Abschnitt über POKEs und andere nützliche Routinen. Ob es um die Programmierung der Funktionstasten, Programme die sich selber starten, „Maus"-Simulation mit dem Joystick oder die Änderung von Speicherbereichen geht, man ist immer wieder über die Fülle der Möglichkeiten erstaunt. Der Clou dieses Buches sind aber die vielen Programmlistings. Die BASIC-Erweiterungen allein stellen schon ein erstklassiges Toolkit dar: APPEND (Anhängen von Programmen, AUTO (*automatische Zeilennummerierung*), BASIC-Befehle auf Tastendruck, PRINT POSITION, UNNEW, Strings größer als 88 Zeichen einlesen und vieles mehr. Die Bandbreite reicht von Spielen wie Goldgräber oder Starshooter bis zu nützlichen Programmen wie Cassetteninhaltsverzeichnis und -katalog mit automatischem Suchen nach Dateien und einem Terminkalender. Für den VC-20 Anwender ist dieser 324 Seiten-Wälzer eine wahre Fundgrube, in der es immer etwas neues zu entdecken gibt.



VC-20 TIPS & TRICKS, 3. erweiterte und überarbeitete Auflage, 1984, 324 Seiten, DM 49,–

Interessant.

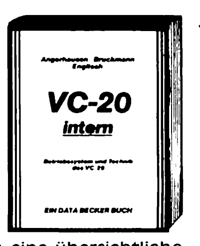
Einen guten Einstieg in PASCAL bietet dieses Trainingsbuch. Es gibt eine leichtverständliche Einführung, sowohl in UCSD-PASCAL wie auch in PASCAL64, wobei allerdings EDV-und BASIC-Grundkenntnisse vorausgesetzt werden. Der Autor, Ottmar Korbmacher, ist Student der Mathematik. Ihm gelingt es, in einem sprachlich aufgelockerten Stil mit vielen interessanten Beispielprogrammen, dem Leser Programmstrukturen, Ein/Ausgabe, Arithmetik und Funktionen, Prozeduren und Rekursionen, Sets, Files und Records näherzubringen. Die Übungsaufgaben am Ende jeden Kapitels helfen dabei, das Gelernte zu vertiefen. Ein Anhang mit allen PASCAL-Schlüsselworten, der ansich schon ein umfangreiches Lexikon darstellt, macht das Buch für jeden PASCAL-Anwender interessant.



DAS TRAININGSBUCH ZU PASCAL, 1984, ca. 250 Seiten, DM 39,–

Bewährt.

Die bereits dritte Auflage von VC-20 INTERN ist wieder erheblich erweitert worden. Das Buch beschäftigt sich ausführlich mit der Technik und dem Betriebssystem des VC-20. Dazu gehört natürlich zuerst einmal ein ausführlich dokumentiertes ROM-Listing. Dazu gehört auch die Belegung der Zeropage, dem wichtigsten Speicherbereich für den 6502-Prozessor, eine übersichtliche Auflistung der Adressen aller Betriebssystemroutinen, ihrer Bedeutung und ihrer Übergabeparameter. Dies ermöglicht dem Programmierer endlich, den VC-20 von Maschinensprache aus sinnvoll einzusetzen. Denn warum Routinen, die bereits vorhanden sind, noch einmal schreiben? Weiterer Inhalt: Einführung in die Maschinensprache – Maschinensprachemonitor, Assembler, Disassembler – Verbindung von Maschinensprache- und BASIC-Programmen – Beschreibung der wichtigen IC's des VC-20 – Blockschaltbild – drei Original COMMODORE-Schaltpläne. Das Buch braucht jeder der sich intensiv mit der Maschinenspracheprogrammierung des VC-20 auseinandersetzen möchte.



VC-20 INTERN, 3. Auflage, 1984, ca. 230 Seiten, DM 49,–

Starthilfe!

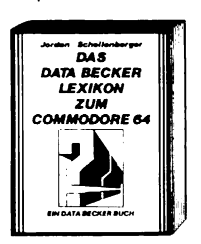


Das sollte Ihr erstes Buch zum COMMODORE 64 sein: 64 FÜR EINSTEIGER ist eine sehr leicht verständliche Einführung in Handhabung, Einsatz, Ausbaumöglichkeiten und Programmierung des COMMODORE 64, die keinerlei Vorkenntnisse voraussetzt. Sie reicht vom Anschluß des Geräts über die Erklärung der einzelnen Tasten und Funktionen sowie die Peripheriegeräte und ihre Bedienung bis zum ersten Befehl. Schritt für Schritt führt das Buch Sie in die Programmiersprache BASIC ein, wobei Sie nach und nach eine komplette Adressenverwaltung erstellen, die Sie anschließend nutzen können. Zahlreiche Abbildungen und Bildschirmfotos ergänzen den Text. Viele Anwendungsbeispiele geben nützliche Anregungen zum sinnvollen Einsatz des COMMODORE 64. Das Buch ist sowohl als Einführung als auch als Orientierung vor dem 64er Kauf gut geeignet.

64 FÜR EINSTEIGER, 1984, ca. 200 Seiten, DM 29,–

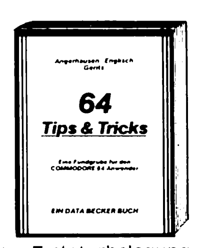
Von A bis Z.

So etwas haben Sie gesucht: Umfassendes Nachschlagewerk zum COMMODORE 64 und seiner Programmierung. Allgemeines Computerlexikon mit Fachwissen von A–Z und Fachwörterbuch mit Übersetzungen wichtiger englischer Fachbegriffe – das DATA BECKER LEXIKON ZUM COMMODORE 64 stellt praktisch drei Bücher in einem dar. Es enthält eine unglaubliche Vielfalt an Informationen und dient so zugleich als kompetentes Nachschlagewerk und als unentbehrliches Arbeitsmittel. Viele Abbildungen und Beispiele ergänzen den Text. Ein Muß für jeden COMMODORE 64 Anwender!



DAS DATA BECKER LEXIKON ZUM COMMODORE 64, 1984, 354 Seiten, DM 49,–

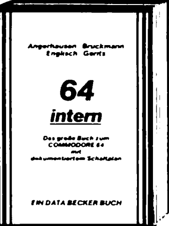
Fundgrube.



64 Tips & Tricks ist eine hochinteressante Sammlung von Anregungen zur fortgeschrittenen Programmierung des COMMODORE 64, POKE's und andere nützliche Routinen, interessanten Programmen sowie interessanten Programmiertips & -tricks. Aus dem Inhalt: 3D-Graphik in BASIC – Farbige Balkengraphik – Definition eines eigenen Zeichensatzes – Tastaturbelegung und ihre Änderung – Dateneingabe mit Komfort – Simulation der Maus mit einem Joystick – BASIC für Fortgeschrittene – C-64 spricht deutsch – CP/M auf dem COMMODORE 64 – Druckeranschluß über den USER-Port – Datenübertragung von und zu anderen Rechnern – Expansion-Port – Synthesizer in Stereo – Retten einer nicht ordnungsgemäß geschlossenen Datei – Erzeugen einer BASIC-Zeile in BASIC – Kassettenpuffer als Datenspeicher – Sortieren von Stringfelder – Multitasking auf dem COMMODORE 64 – POKE's und die Zeropage – GOTO, GOSUB und RESTORE mit berechneten Zeilennummern, INSTR und STRING-Funktion – Repeat-Funktion für alle Tasten – und vieles andere mehr. Alle Maschinenprogramme mit BASIC-Ladeprogrammen. 64 Tips & Tricks ist eine echte Fundgrube für jeden COMMODORE 64 Anwender. Schon über 65000mal verkauft!

64 TIPS & TRICKS, 1984, über 300 Seiten, DM 49,–

Know-how!



350 Seiten dick ist die 4. erweiterte und überarbeitete Auflage von 64 INTERN geworden. Das bereits über 65000mal verkaufte Standardwerk bietet jetzt noch mehr Informationen. Hinzugekommen ist ein Kapitel über den IEC-Bus und viele, viele Ergänzungen, die sich im Laufe der Zeit angesammelt haben. Ebenfalls überarbeitet und noch ausführlicher ist jetzt die Dokumentation des ROM-Listings. Weitere Themen: genaue Beschreibung des Sound- und Video-Controllers mit vielen Hinweisen zur Programmierung von Sound und Grafik, der Ein/Ausgabesteuerung (CIAs), BASIC-Erweiterungen (RENEW, HARDCOPY, PRINTUSING), Hinweise zur Maschinenprogrammierung wie Nutzung der E/A-Routinen des Betriebssystems, Programmierung der Schnittstelle RS 232, ein Vergleich VC20 – C-64 – CBM zur Umsetzung von Programmen. Dies und viele weitere Informationen machen das umfangreiche Werk zu einem unentbehrlichen Arbeitsmittel für jeden, der sich ernsthaft mit Betriebssystem und Technik des C-64 auseinandersetzen will. Zum professionellen Gehalt des Buches tragen auch zwei Original-COMMODORE-Schaltpläne zum Ausklappen und zahlreiche ausführlich beschriebene und dokumentierte Fotos, Schaltbilder und Blockdiagramme bei.

64 INTERN, 4. überarbeitete und erweiterte Auflage, 1984, ca. 350 Seiten, DM 69,–

Erfolgreich.



64 für Profis zeigt, wie man erfolgreich Anwendungsprobleme in BASIC löst und verrät die Erfolgsgeheimnisse der Programmierprofis. Vom Programmentwurf über Menüsteuerung, Maskenaufbau, Parametrisierung, Datenzugriff und Druckausgabe bis hin zur guten Dokumentation wird anschaulich mit vielen Beispielen dargestellt wie Profi-Programmierung vor sich geht. Besonders stolz sind wir auf die völlig neuartige Datenzugriffsmethode QUISAM, die in diesem Buch zum ersten Mal vorgestellt wird. QUISAM erlaubt eine beliebige Datensatzlänge, die dynamisch mit der Eingabe der Daten wächst. Eine lauffertige Literaturstellenverwaltung veranschaulicht die Arbeitsweise von QUISAM. Neben diesem Programm finden Sie noch weitere Programme zur Lager- und Adressenverwaltung, Textverarbeitung und einen Reportgenerator. Alle diese Programme sind mit Variablenliste versehen und ausführlich beschrieben. Damit sind diese für Ihre Erweiterungen offen und können von Ihnen an Ihre persönlichen Bedürfnisse angepaßt werden. Steigen Sie in die Welt der Programmierprofis ein.

64 FÜR PROFIS, 2. Auflage, 1984, ca. 300 Seiten, DM 49,–

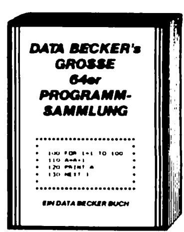
Rundum gut!



Endlich ein Buch, das Ihnen ausführlich und verständlich die Arbeit mit der Floppy VC-1541 erklärt. Das große Floppybuch ist für Anfänger, Fortgeschrittene und Profis gleichermaßen. interessant. Sein Inhalt reicht von der Programmspeicherung bis zum DOS-Zugriff, von der sequentiellen Datenspeicherung bis zum Direktzugriff, von der technischen Beschreibung bis zum ausführlich dokumentierten DOS-Listing, von den Systembefehlen bis zur detaillierten Beschreibung der Programme auf der Test-Demo-Diskette. Exakt beschriebene Beispiel- und Hilfsprogramme ergänzen dieses neue Superbuch. Aus dem Inhalt: Speichern von Programmen – Floppy-Systembefehle – Sequentielle Datenspeicherung – relative Datenspeicherung – Fehlermeldungen und ihre Ursachen – Direktzugriff – DOS-Listing der VC-1541 – BASIC-Erweiterungen und Programme – Overlaytechnik – Diskmonitor – IEC-Bus und serieller Bus – Vergleich mit den großen CBM-Floppies. Ein Muß für jeden Floppy-Anwender! Bereits über 45.000mal verkauft.

DAS GROSSE FLOPPY-BUCH, 2. überarbeitete Auflage, 1984, ca. 320 Seiten, DM 49,–

Füttern erwünscht!

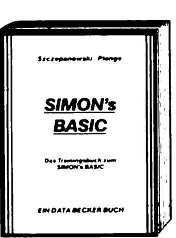


Diese beliebte umfangreiche Programmsammlung hat es in sich. Über 50 Spitzenprogramme für den COMMODORE 64 aus den unterschiedlichsten Bereichen, von attraktiven Superspielen (Senso, Pengo, Master Mind, Seeschlacht, Poisson Square, Memory) über Grafik- und Soundprogramme (Fourier 64, Akustograph, Funktionsplotter) und mathematische Programme (Kurvendiskussion, Dreieck) sowie Utilities (SORT, RENUMBER, DISK INIT, MENUE) bis hin zu kompletten Anwendungsprogrammen wie „Videothek", „File Manager" und einer komfortablen Haushaltsbuchführung, in der fast professionell gebucht wird. Der Hit zu jedem Programm sind aktuelle Programmiertips und Tricks der einzelnen Autoren zum Selbermachen. Also nicht nur abtippen, sondern auch dabei lernen und wichtige Anregungen für die eigene Programmierung sammeln.

DATA BECKER's GROSSE 64er PROGRAMMSAMMLUNG, 1984, 250 Seiten, DM 49,–

Bestseller aus bester Hand

BASIC-PLUS.



SIMON's BASIC ist ein Hit – wenn man es richtig nutzen kann. Auf über 300 Seiten erklärt Ihnen das DATA BECKER Trainingsbuch detailliert den Umgang mit den über 100 Befehlen des SIMON's BASIC. Alle Befehle werden ausführlich dargestellt, auch die, die nicht im Handbuch stehen! Natürlich zeigen wir auch die Macken des SIMON's BASIC und geben wichtige Hinweise wie man diese umgeht. Natürlich enthält das Buch viele Beispielprogramme und viele interessante Programmiertricks. Weiterer Inhalt: Einführung in das CBM-BASIC 2.0 – Programmierhilfen – Fehlerbehandlung – Programmschutz – Programmstruktur – Variablen – Zahlenbehandlung – Eingabekontrolle – Ein/Ausgabe Peripheriebefehle – Graphik – Zeichensatzerstellung – Sprites – Musik – SIMON's BASIC und die Verträglichkeit mit anderen Erweiterungen und Programmen. Dazu ein umfangreicher Anhang. Nach jedem Kapitel finden Sie Testaufgaben zum optimalen Selbststudium und zur Lernerfolgskontrolle.

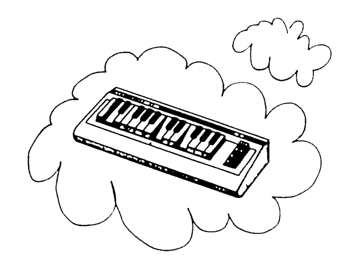
DAS TRAININGSBUCH ZUM SIMON's BASIC, 2. überarbeitete Auflage, 1984, ca. 380 Seiten, DM 49,–

Schrittmacher.



Eine leicht verständliche Einführung in die Maschinenspracheprogrammierung für alle, denen das C-64 BASIC nicht mehr ausreicht. Sie lernen Aufbau und Arbeitsweise des 6510-Mikroprozessors kennen und anwenden. Dabei werden die Analogien zu BASIC Ihnen beim Verständnis helfen. Ein weiteres Kapitel beschäftigt sich mit der Eingabe von Maschinenprogrammen. Dort erfahren Sie auch alles über Monitor-Programme sowie über Assembler. Zum einfachen und komfortablen Erstellen Ihrer eigenen Maschinensprache enthält das Buch einen kompletten ASSEMBLER, damit Sie gleich von Anfang an komfortabel und effektiv programmieren können. Weiterhin finden Sie dort einen DISASSEMBLER, mit dem Sie sich Ihre Maschinenprogramme oder die Routinen des BASIC-Interpreters und des BASIC-Betriebssystems ansehen können. Ein besonderer Clou ist ein in BASIC geschriebener Einzelschrittsimulator, mit dem Sie Ihre Programme schrittweise ausführen können. Dabei werden Sie nach jedem Schritt über Registerinhalte und Flags informiert und können den logischen Ablauf Ihres Programmes verfolgen. Eine unschätzbare Hilfe, besonders für den Anfänger. Als Beispielprogramm finden Sie ausführlich beschriebene Routinen zur Grafikprogrammierung und für BASIC-Erweiterungen. Natürlich sind alle Beispiele und Programme auf den C-64 zugeschnitten.

DAS MASCHINENSPRACHEBUCH ZUM COMMODORE 64, ca. 200 Seiten, DM 39,–



SYNTHIMAT

SYNTHIMAT verwandelt Ihren COMMODORE 64 in einen professionellen, polyphonen, dreistimmigen Synthesizer, der in seinen unglaublich vielen Möglichkeiten großen Systemen kaum nachsteht.

SYNTHIMAT in Stichworten:
drei Oszillatoren (VCOs) mit 7 Fußlagen und 8 Wellenformen – drei Hüllkurvengeneratoren (ADSRs) – ein Filter (VCF) mit 8 Betriebsarten und Resonanzregulierung – VCF mit Eingang für externe Signalquelle – ein Verstärker (VCA) – Ringmodulation mit allen drei VCOs – 8 softwaremäßig realisierte Oszillatoren (LFOs) – kräftiger Klang durch polyphones Spielen – zwei Manuale (Solo und Begleitung) – speichern von bis zu 256 Klangregistern – schneller Registerwechsel – speichern von 9 Registerdateien auf Diskette – „Bandaufnahme" auf Diskette durch direktes Spielen – keine lästige Noteneingabe – speichern von bis zu 9 „Bandaufnahmen" je Diskette – integrierte 24 Stunden-Echtzeituhr – einstellbares PITCH-BENDING – farblich gekennzeichnete, übersichtlich angeordnete Module – umfangreiches Handbuch – läuft mit einem Diskettenlaufwerk – Diskettenprogramm.

DM 99,–



STRUKTO 64

STRUKTO 64 ist eine fantastische neue Programmiersprache für strukturiertes Programmieren mit dem C-64 und für alle Programmierer geeignet, die den C-64 als Allround-Computer einsetzen und auf einfache Weise anspruchsvolle Programme erstellen wollen.

STRUKTO 64 in Stichworten:
Interpretersprache, die die Vorzüge von BASIC und PASCAL vereint – strukturiertes Programmieren – übersichtliche Programme – leichte Erlernbarkeit – einfache Bedienung – eingebautes Toolkit erleichtert das Eingeben und Verbessern von Programmen – leichteres Arbeiten mit der Floppy – Sprite-Editor ermöglicht das Einlesen der Sprite-Formen direkt vom Bildschirm – Graphikbedienung wird mit gut durchdachten Befehlen unterstützt – Abspielen von Musik ist unabhängig vom Programmablauf möglich – ca. 80 neue Befehle – lieferbar als Diskettenprogramm – ausführliches deutsches Handbuch.

DM 99,–



Für viele ein Traum, für die meisten bisher zu teuer: die Rede ist von einer echten Datenbank für den 64er. SUPERBASE 64 füllt eine Lücke. Nicht allein die Kapazität, die verwaltet werden kann, bewegt sich in professionellen Regionen, die ausgeprägten Fähigkeiten des SUPERBASE 64 im Rechnen und Kalkulieren lassen dieses Paket beinahe als Rund-Um-Software erscheinen.

SUPERBASE 64 in Stichworten:
maximale Datensatzlänge 1108 Zeichen, verteilt auf bis zu 4 Bildschirmseiten – bis zu 127 Felder pro Datensatz, wobei Textfelder bis zu 255 Zeichen lang sein können – insgesamt 15 Einzeldateien können zu einer SUPERBASE-Datenbank verknüpft werden – Speicherkapazität nur durch Diskette begrenzt – umfangreiche Auswertungsmöglichkeiten und komfortabler Report-Generator – Kalkulationsmöglichkeiten und Rechnen – Import- (Einlesen von externen Daten) und Export- (Ausgabe von SUPERBASE Dataien als sequentielle Datei) Funktionen ermöglichen Datenaustausch mit anderen Programmen – durch leistungsfähige, eigene Datenbanksprache auch als kompletter Anwendungsgenerator verwendbar.

DM 398,–



MASTER 64

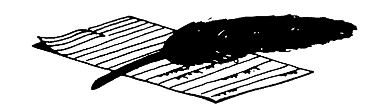
MASTER 64 ist ein professionelles Programmentwicklungssystem für den C-64, das es Ihnen ermöglicht, die Programmentwicklungszeit auf einen Bruchteil der sonst üblichen Zeit zu reduzieren. MASTER 64 bietet einen Programmkomfort, den Sie nutzen sollten.

MASTER 64 in Stichworten:
70 zusätzliche Befehle – Bildschirmmaskengenerator – definieren von Bildschirmzonen – Eingabe aus Zonen – formatierte Ausgabe – Abspeicherung von Bildschirminhalten – Arbeiten mit mehreren Bildschirmmasken – ISAM Dateiverwaltung, in der Datensätze über einen Zugriffschlüssel angesprochen werden können – Datensätze bis zu 254 Zeichen – Schlüssellänge bis zu 30 Zeichen – Dateigröße nur von Diskettenkapazität abhängig – Zugriff über Schlüssel und Auswahlmasken – Bildschirm- und Druckmaskengenerator – Erstellung beliebiger Formulare und Ausgabemasken – BASIC-Erweiterungen – Toolkitfunktionen – Mehrfachgenaue Arithmetik (Rechnen mit 22 Stellen Genauigkeit).

DM 198,–

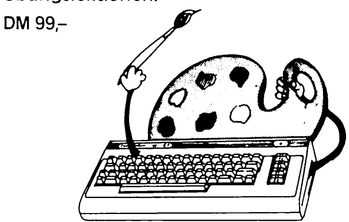
TEXTOMAT

Das Bearbeiten von Texten gehört zum wichtigsten Betätigungsfeld von Homecomputer-Anwendern. So ist es auch nicht verwunderlich, daß eine Unzahl verschiedener Textprogramme für den 64er angeboten wird. TEXTOMAT zeichnet sich dadurch aus, daß er auch vom Einsteiger sofort benutzt werden kann. Über eine Menuezeile können alle Funktionen angewählt werden. Selbstverständlich beherrscht TEXTOMAT deutsche Umlaute und Sonderzeichen.



TEXTOMAT in Stichworten:
Diskettenprogramm – durchgehend menuegesteuert – deutscher Zeichensatz auch auf COMMODORE-Druckern Rechenfunktionen für alle Grundrechenarten – 24.000 Zeichen pro Text im Speicher – beliebig lange Texte durch Verknüpfung – horizontales Scrolling für 80 Zeichen pro Zeile – läuft mit 1 oder 2 Floppies – frei programmierbare Steuerzeichen – Formularsteuerung für Randeinstellung u.s.w. – komplette Bausteinverarbeitung – Blockoperationen, Suchen und Ersetzen – Serienbriefschreibung mit DATAMAT – formatierte Ausgabe auf Bildschirm – an fast jeden Drucker anpaßbar – ausführliches deutsches Handbuch mit Übungslektionen.

DM 99,–

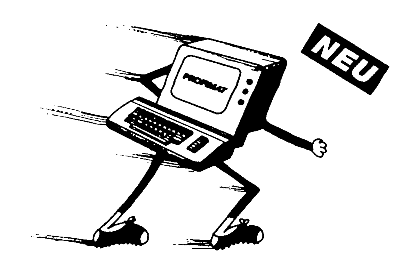


PAINT PIC

Malen (!) mit dem Computer, welch eine faszinierende Idee. Mit dem Malprogramm PAINT PIC für den COMMODORE 64 wird diese Idee Realität. Mit PAINT PIC ist es auch für den Einsteiger leicht, fantastische Computerbilder zu erstellen. Man kann die Bilder auf Diskette abspeichern und wieder laden und selbstverständlich steht auch weiterhin der COMMODORE-Zeichensatz zur Verfügung. Wichtig: PAINT PIC benötigt keine zusätzliche Hardware.

PAINT PIC in Stichworten:
Programmsteuerung: Tastatur – Steuerung des Stifts: Cursortasten und eckige Klammer (diag.) (Joystick kann benutzt werden) – Routinen: Linien, Rechtecke, Dreiecke, Parallelogramme, Kreise, Kreisbögen, Ellipsen, Bestimmung von Mittelpunkt, und perspektivischer Linie, Kopieren und Drehen von Teilbildern, Verdoppeln, halbieren und spiegeln von Teilbildern – Modi: Malstiftmodus (schmale Linie) Pinselmodus (8 verschiedene Breiten) (Art der Linie selbst definierbar) — Textmodus (kompl. Zeichensatz COMMODORE) (Hoch-Tiefschrift) – Speichern: Teilbilder (Blöcke) oder ganze Bilder — Menue: 1 Hauptmenue mit 8 Untermenues – mit ausführlichem deutschen Handbuch – Diskettenprogramm – Bilder kann man auf Diskette abspeichern.

DM 99,–

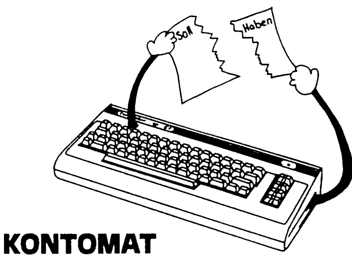


PROFIMAT

Wer sich tiefer in die Innereien des Computers begeben will, kommt ohne besonderes Werkzeug nicht aus. Einerseits muß der volle Einblick in alle Speicherbereiche möglich sein, andererseits soll der Umgang mit Maschinenprogrammen so komfortabel wie möglich gestaltet sein. PROFIMAT hat Lösungen für beide Probleme: Der Maschinensprache-Monitor PROFI-MON bietet alle Hilfsmittel zum Umgang mit Maschinenprogrammen; PROFI-ASS ist ein Macro-Assembler, der das Schreiben von Maschinenprogrammen fast so einfach macht wie das Programmieren in BASIC.

PROFIMAT in Stichworten:
Registerinhalte und Flags anzeigen – Speicherinhalte anzeigen – Maschinenprogramme laden, ausführen und speichern – Speicherbereiche durchsuchen, vergleichen, füllen und verschieben – echter Einzelschrittmodus – Setzen von Unterbrechungspunkten – schneller Trace-Modus – Rückkehr zu BASIC – formatfreie Eingabe – Verkettung beliebig vieler Quellprogramme – erzeugter Objektcode kann in Speicher oder auf Diskette gehen – formatiertes Assemblerlisting – ladbare Symboltabellen – redefinierbare Symbole – Operatoren – Unterstützung der Fließkommaarithmetik – bedingte Assemblierung – Assemblerschleifen – MACROS mit beliebigen Parametern.

DM 99,–

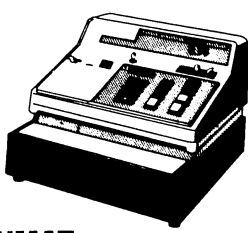


KONTOMAT

KONTOMAT ist ein menuegesteuertes Einnahme-Überschußprogramm nach § 4(3) EStG mit Kassenbuch, Bankkontenüberwachung, automatischer Steuerbuchung, AFA Tabellenerstellung, Kontenblättern, Ermittlung der USt.-Voranmeldungswerte und Monats- und Jahresabrechnung. Der neue KONTOMAT ist voll parameterisiert und läßt sich damit an Ihre Bedürfnisse anpassen. Für alle Gewerbetreibenden, die nicht laut HGB zur Buchführung verpflichtet sind. KONTOMAT ist für den gewerblichen Einsatz, aber auch als Lernprogramm oder zur Haushaltsbuchführung geeignet.

KONTOMAT in Stichworten:
Diskettenprogramm – maximal 120 Konten – Beträge mit bis zu 6 Vor- und 2 Nachkommastellen – 4 Mehrwert- und Vorsteuersätze – intervallmäßige Belegeingabe – 4 Buchungsarten (SOLL, HABEN, SOLL/HABEN und HABEN/SOLL) – Anzeige der Soll- und Habensumme bei mehrfachen Buchungssätzen – komfortable Belegeingabe mit Datum, Buchungstext, Stuerkennzeichen und Betrag – Druck des Journals während der Belegeingabe – Druck von umfangreichen Kontenblättern – Druck einer Summen- und Saldenliste mit Monats- und Jahresumsatzsummen – betriebswirtschaftliche Auswertung mit Druckausgabe – Ermittlung und Druckausgabe der Umsatzsteuerzahllast – Speicherung der Anlagegüter und automatische Abschreibung am Jahresende – übersichtliche AfA-Liste – arbeitet mit 1 oder 2 Laufwerken – umfangreiches deutsches Handbuch.

DM 148,–

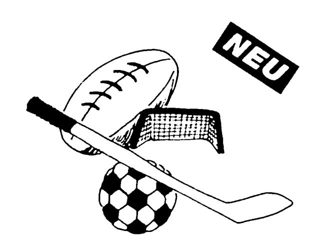


FAKTUMAT

Mit FAKTUMAT ist das Schreiben von Rechnungen kein Alptraum mehr. Eine Sofortfakturierung mit integrierter Lagerbuchführung. Individuelle Anpassung von Steuersätzen, Maßeinheiten und Firmendaten. Kunden- und Artikelstamm voll pflegbar. Schneller Zugriff auf Kunden- und Artikeldaten, über freidefinierbaren, 6-stelligen Schlüssel. Automatische Fortschreibung von Artikel- und Kundendaten, individuell nutzbar. Alles in allem die Arbeits- und Zeitersparnis, die Sie sich schon immer gewünscht haben.

FAKTUMAT in Stichworten:
voll menuegesteuert – läuft mit einer oder zwei Floppies – Diskettenwechsel (eine Floppy) nur beim Wechsel vom Hauptmenue ins Unterprogramm und umgekehrt – mit Ausnahme des Ausschaltens der Floppy während der Verarbeitung werden alle Fehler abgefangen (z. B. Drukker nicht eingeschaltet – arbeitet mit 1525, 1526 (?), MPS 801, EPSON Drucker und DATA BECKER Interface – voll parameterisiert: Firmenkopf, MWSt. und Rabattsätze, Größe der Dateien beliebig wählbar – 5 Zeilen für Firmenkopf je 30 Zeichen (erste Zeile erscheint auf der Rechnung in Breitschrift – 4 Mehrwertsteuer-Sätze; während der Rechnungsschreibung können also Artikel mit unterschiedlichem Mehrwertsteuer-Satz verrechnet werden – 10 Rabattsätze (Rabattsatz 1 vorbelegt mit 0%), bei der Rechnungsschreibung kann jedem Artikel ein Rabattsatz zugewiesen werden – maximal 1900 Artikel bei 50 Kunden oder 950 Kunden bei 100 Artikel (max. Artikel = [1000-Kunden]*2; max. Kunden = [2000-Artikel]/2) – manuelle Eingabe von Artikeln und/oder Kunde während der Rechnungsschreibung – d. h. es können mehr Artikel verrechnet weden als überhaupt in die Datei passen (bei Verzicht auf Lagerbuchführung) bzw. es können Rechnungen an Kunden geschrieben werden, die nicht erfaßt wurden – integrierte Lagerbuchführung mit Ausgabe einer Inventurliste – Rechnungsbeträge und Datum werden in der Kundendatei festgehalten – Druck von: Rechnung (mit Abbuchen aus Lager), Rechnung (ohne Abbuchen aus Lager), Lieferschein – deutsches detailliertes Handbuch mit Übungs- und Anwendungsteil – deutsche Bedienerführung innerhalb des Programms (z. B. „Artikel nicht vorhanden" anstelle „RECORD NOT PRESENT").

DM 148,–

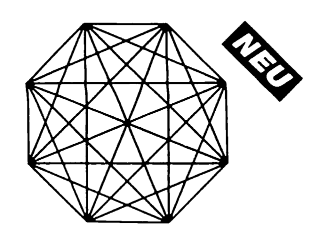


UNI-TAB

Heute schon die Bundesliga-Tabelle von morgen kennen, das geht mit UNI-TAB. Alle Rechnereien, die man ohne dieses Programm nie machen würde, lassen sich in Sekundenschnelle durchführen. Wer will, kann mit simulierten Spielergebnissen den Weltmeister '86 vorausberechnen. Aber nicht nur Fußball-Ligen können tabellarisch erfaßt werden, fast alle Sportarten sind UNI-TAB-fähig. Gag am Rande: für viele Sportarten stehen die bekannten Piktogramme zur Verfügung.

UNI-TAB in Stichworten:
Menuesteuerung über die Funktionstasten mit leicht verständlichen Auswahlmöglichkeiten – Bedienerfreundlich (Mannschaften werden über Kennzahlen gesteuert) – Ligen mit 4 bis 20 Mannschaften können verwaltet werden (6 bis 38 Spieltage möglich) – unsinnige Ligen (z. B. 13 Mannschaften sollen 5 Spieltage absolvieren) sind ausgeschlossen – favorisierte Mannschaft kann während des Programmablaufs durch reverse Darstellung gekennzeichnet werden – Tabelle kann geändert werden (wichtig bei Spielanullierungen) – drei verschiedene Tabellenarten können abgespeichert und später eingelesen werden (die aktuelle Tabelle [unabhängig von der Vollständigkeit eines Spieltages], der komplette Spieltag [Vollständigkeit und Nummer des Spieltages werden automatisch errechnet], die simulierte Tabelle [der Anwender kann so selbst Schicksal spielen und seinen Tip später mit dem tatsächlichen Geschehen vergleichen]) – zwei verschiedene Arten der Saisonübersicht (die statistische Übersicht zeigt an, welchen Tabellenplatz das jeweilige Team bei welchem Punkte- und Torverhältnis an den einzelnen Spieltagen einnahm; die graphische Übersicht zeigt die Leistungskurve jeder Mannschaft) – alle Tabellen und Graphiken sind als Hardcopy auf einem Drukker darstellbar – bei Fehlbedienung (z.B. gewünschte Druckausgabe bei nicht eingeschaltetem Drucker) erscheinen leicht verständliche deutsche Fehlermeldungen.

DM 69,–



SUPERGRAFIK 64

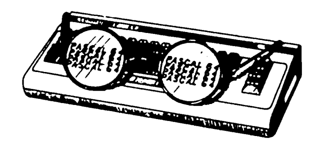
Entdecken Sie die faszinierende Welt der Computergraphik mit SUPERGRAFIK 64, der starken Befehlserweiterung mit den vielseitigen Möglichkeiten. Durch die neue verbesserte Version jetzt noch leistungsstärker.

SUPERGRAFIK 64 in Stichworten:
2 unabhängige Graphikseiten (320 x 200 Punkte) – logische Verknüpfung der beiden Graphikseiten (AND, OR, EXOR) – 1 Standard Low-Graphik Seite (80 x 50 Punkte) – Normalfarben Graphik (300 x 200 Punkte) – Multicolor-Graphik (160 x 200 Punkte) – verdecktes Zeichnen (z. B. Text sichtbar, Graphikseite 2 wird erstellt) – Textfenster in der Graphik – 183 Befehle und Befehlskombinationen (1. Für jeden Befehl wählbare Zwischenmodi: Zeichnen, Löschen, Punktieren, Graphik-Cursor bewegen, Zeichnen mit/ohne Farbsetzung, Punkte zählen; 2. Durch einfache Befehle zu steuernde Graphikfiguren: Punkt, Linie, Linienschar, Linie vom Graphik-Cursor, Kreise, Kreisbögen, Ellipse, Ellipsenbögen, selbstdefinierbare Figuren, rotieren und vergrößern dieser Figuren, Rahmen, Feld, Text in Graphik; 3. Weitere Graphikbefehle: Graphikseiten- und Moduswechsel, Graphik löschen, Graphik invertieren, Scrolling von Text und Graphik, Wählen der Rahmen-, Hintergrund-, Zeichen- oder Punktfarbe) – Speichern, Laden von Graphik (auch verdeckt) – Kopieren des Textbildschirms in die Graphikseite – Hardcopies für EPSON, Seikosha GP100VC, Farb(!)drucker Seikosha GP700 und andere mit DATA BECKER Interface – 16! Sprites gleichzeitig auf dem Bildschirm – alle Sprite-Eigenschaften veränderbar – Positionieren und Bewegen (!) von 16 Sprites gleichzeitig und unabhängig voneinander, während das übrige Programm weiterläuft (IRQ) – Sprite-Kollisionsüberprüfung, Joystickunterstützung – automatische Unterbrechung des BASIC-Programms bei Kollisionen (Interrupt), Sprung in Unterbrechungsroutine, dann Weiterführung des Hauptprogramms – komfortable Soundprogrammierung mit Verstellung aller möglichen Sound-Parameter (Lautstärke, Klang, Filter, Tonhöhe, Tonlänge), ebenfalls unabhängig vom übrigen Programmlauf – zahlreichen Programmiertools (MERGE, RENUMBER usw.) – umfangreiche Anleitung – Diskettenprogramm.

DM 99,–

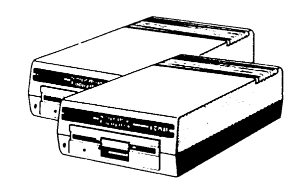
PASCAL 64

Beim Wort „Compiler" fällt dem Eingeweihten sicher der Begriff „Geschwindigkeit" ein. Ein PASCAL-Compiler sollte jedoch weitere Assoziationen wecken. Strukturiertes Programmieren heißt das Zauberwort. PASCAL wurde eigens zu didaktischen Zwecken entwickelt und erfüllt diese Aufgabe auch heute noch. Der PASCAL 64 Compiler bringt diese phantastische Programmiersprache auf den 64er.
Gerade die neue, verbesserte Version unterstützt die Möglichkeiten des C-64 in jeder Hinsicht und macht leistungsfähige Programme möglich.



PASCAL 64 in Stichworten:
besitzt einen sehr umfangreichen Befehlsvorrat – erlaubt Interruptprogrammierung und bietet Schnittstellen zu Monitor und Assembler – erzeugt sehr schnelle Programme in reinem Maschinencode – unterstützt relative Dateiverwaltung, Graphik und Sound – bietet die Datentypen REAL, INTEGER, CHAR und BOOLEAN sowie Aufzähltypen und POINTER, die zu Datenstrukturen RECORD, SET, ARRAY und PACKED ARRAY kombiniert werden können – erlaubt vorzeitigen Abschluß von Prozeduren mit EXIT, uneingeschränkte Rekursionen und komfortable Verarbeitung von Teilfeldern (Strings) – ist ein ausgereiftes, deutsches Produkt und wird mit ausführlichem Handbuch geliefert.

DM 99,–

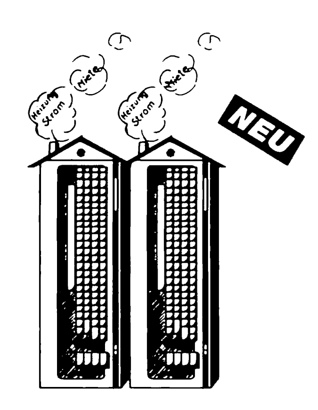


DISKOMAT

Der Umgang mit Diskettenlaufwerken ist für viele noch immer mit Geheimnissen belastet. Andere stören sich an den wenig komfortablen Diskettenbefehlen des BASIC V2. DISKOMAT bringt Abhilfe; alle Diskettenbefehle des BASIC 4.0 stehen zur Verfügung. Außerdem können mit dem Programm SUPERTWIN zwei 1541-Laufwerke wie ein Doppellaufwerk verwaltet werden. Für Benutzer, die sich die Fähigkeiten der Floppy 1541 ganz erschließen wollen, steht der DISK-MONITOR bereit; er macht es endlich möglich, den direkten Zugriff auf einzelne Blocks einfach und bequem vorzunehmen.

DISKOMAT in Stichworten:
Diskettenprogramm – DISK BASIC unterstützt Diskettenbefehle des BASIC 4.0 (CONCAT, HEADER, APPEND, RENAME, OPEN, COLLECT, DSAVE, SCRATCH, DCLOSE, BACKUP, DLOAD, DIRECTORY, RECORD, COPY, CATALOG, DS & DS$) – SUPER TWIN behandelt 2 Laufwerke 1541 wie ein Doppellaufwerk – DISK-MONITOR ermöglicht direkte Analyse und Manipulation von Disketten (direktes Lesen und Schreiben einzelner Blöcke, ändern von Blöcken mittels Bildschirm-Editor, Anzeige des Diskettenstatus, direktes Absenden von Disketten-Befehlen) – ausführliches deutsches Handbuch beschreibt jeden einzelnen der 3 Programmteile.

DM 99,–



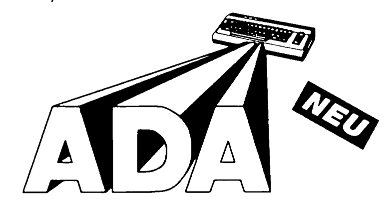
HAUSVERWALTUNG

Jetzt können alle Hausbesitzer aufatmen: das Programm HAUSVERWALTUNG bietet ihnen eine sehr komfortable Verwaltung der Mietwohnungen mit dem COMMODORE 64.
Alles, was Sie dazu brauchen, ist ein COMMODORE 64, ein Diskettenlaufwerk 1541, ein anschlußfähiger Drucker und das obengenannte Programm HAUSVERWALTUNG. Die nachfolgenden und viele weitere leistungsfähige Features ermöglichen eine äußerst rationelle Verwaltung Ihrer Mietwohnungen.

HAUSVERWALTUNG in Stichworten:

Dikettenprogramm – Verwaltung von 50 Einheiten pro Objekt möglich – Stammdatenverwaltung für Häuser und Mieter – Verbuchen der Miete, Nebenkosten und Garagenmieten – Mietkontoanzeige – Haus- und Mieteraufstellung – Mahnungen – Verbuchen der anfallenden Kosten – Kostengegenüberstellung – Jahresendabrechnung mit automatischem Jahresübertrag – umfangreiches deutsches Handbuch.

DM 198,–



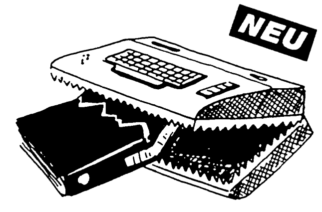
TRAININGSKURS zu ADA

Diese Programmiersprache der Zukunft, die das Pentagon in Auftrag gegeben hat, wird jetzt durch DATA BECKER auch dem C-64 Anwender zugänglich gemacht durch den TRAININGSKURS zu ADA, der eine sehr gute Einführung in diese Supersprache bietet. Der dazu gelieferte Compiler liefert ein umfangreiches Subset der Sprache.

ADA in Stichworten:

blockstrukturierte Programme – modularer Aufbau der Programme – ermöglicht die Behandlung von Ausnahmezuständen – Fehlerüberprüfung beim Übersetzen und zur Laufzeit – ermöglicht das einfache Einbinden von Maschinenprogrammen – sehr leichtes Arbeiten mit Programmbibliotheken – Programmdiskette enthält Editor, Übersetzer, Assembler und Disassembler – umfangreiches deutsches Handbuch.

DM 198,–



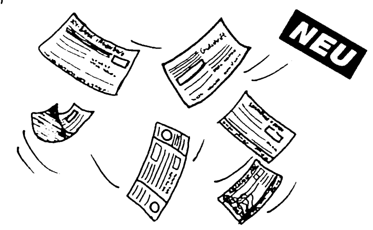
DATAMAT

Daten verwalten kann ein schier endloses Hantieren mit Karteikästen und Aktenordnern bedeuten; kann aber auch C-64 plus DATAMAT heißen. Dann wird Suchen und Sortieren zum Spaß. Der DATAMAT bietet in seiner neuen Version einiges, was in dieser Preisklasse bisher unvorstellbar schien. Nicht nur Geschwindigkeit und Bedienungsfreundlichkeit wurden weiter verbessert, auch die Anpassung an die meisten Drucker ist inzwischen machbar.

DATAMAT in Stichworten:

menuegesteuertes Diskettenprogramm, dadurch extrem einfach zu bedienen – für jede Art von Daten – völlig frei gestaltbare Eingabemaske – 50 Felder pro Datensatz – 253 Zeichen pro Datensatz – bis zu 2000 Datensätze pro Datei je nach Umfang – Schnittstelle zu TEXTOMAT – läuft mit 1 oder 2 Floppies – völlig in Maschinensprache – extrem schnell – deutscher Zeichensatz auch auf COMMODORE-Druckern – fast jeder Drucker anschließbar – ausdrucken über RS 232 – duplizieren der Datendiskette – verbesserte Benutzerführung – Hauptprogramm komplett im Speicher (kein Diskettenwechsel mehr) – integrierte Minitextverarbeitung – deutsches Handbuch mit Übungslektionen
Sie können:
jeden Datensatz in 2 – 3 Sekunden suchen – nach beliebigen Feldern selektieren – nach allen Feldern gleichzeitig sortieren – Listen in völlig freiem Format drucken – Etiketten drucken.

DM 99,–



ZAHLUNGSVERKEHR

Umfangreicher Zahlungsverkehr kann zur Plage werden. Das Software-Paket ZAHLUNGSVERKEHR übernimmt den größten Teil dieser Arbeit. Außer den notwendigen Fähigkeiten für das Ausfüllen und Auflisten von Überweisungen und Schecks ist der ZAHLUNGSVERKEHR in der Lage, Sammellisten, Einzugslisten etc. selbständig zusammenzustellen.

ZAHLUNGSVERKEHR in Stichworten:

Diskettenprogramm – max. 100 Zahlungsempfänger pro Diskette – drei definierbare Absenderbanken – 25 Zahlungsdateien – 14 frei definierbare Formulare – Kontrolldruck bei Belegeingabe möglich – Eingabe von Rechnungsdaten oder eines Verwendungszwecks – Ausdruck einer Sammel-Überweisungsliste – Korrekturmöglichkeit der einzelnen Zahlungsdateien – arbeitet mit einer oder zwei Floppies – umfangreiches deutsches Handbuch.

DM 148,–

DAS STEHT DRIN:

Das CASSETTENBUCH enthält alles über die Datassette und Cassettenspeicherung mit Ihrem COMMODORE 64 und VC-20. Mit absoluten Spitzenprogrammen.

Aus dem Inhalt:

- Befehle zur Handhabung der Datassette
- Lautsprecher an der Datassette, Kopfjustage
- Steuerung der Datassette per Programm
- Anhängen von Basicprogrammen
- Sekundäradresse, Statusvariable
- Speicherformat von Daten und Programmen
- Wichtige Speicherstellen des Cassettenbetriebssystems
- „FastTape"-Betriebssystem macht die Datassette 10 – 20 mal schneller
- Komfortables Datenverwaltungsprogramm
- Backup von Disk auf Cassette und umgekehrt

. . . und vieles mehr!

UND GESCHRIEBEN HAT DIESES BUCH:

Dirk Paulissen ist Radio- und Fernsehtechniker, Student und vor allem ein Programmierer, der 64er und VC-20 in- und auswendig kennt. Nach VC-20 Tips & Tricks ist dies bereits sein zweites DATA BECKER Buch, in dem er es versteht, komplizierte Sachverhalte leicht verständlich darzustellen.

ISBN 3-89011-030-4